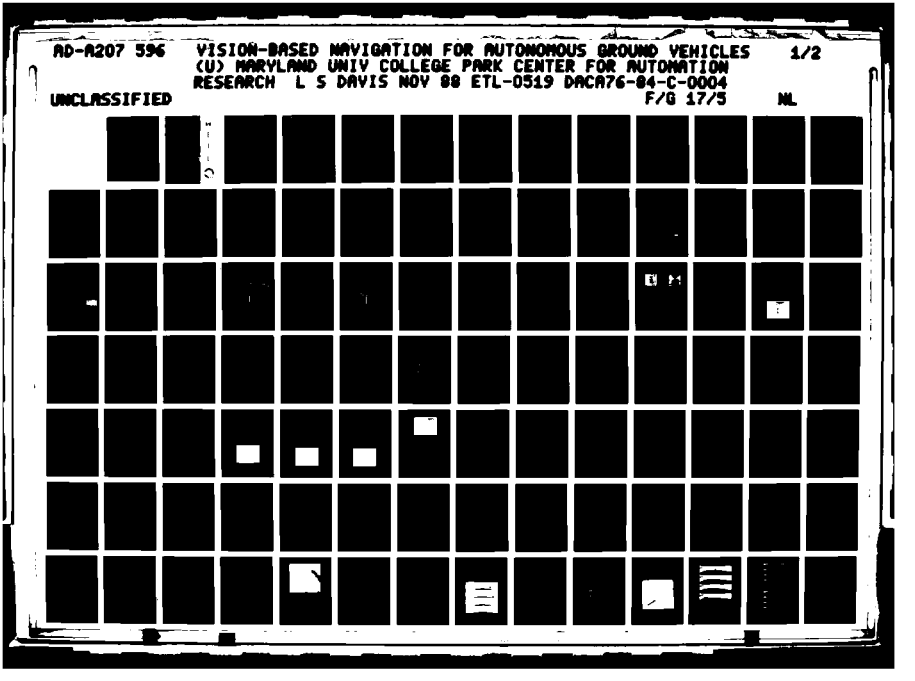
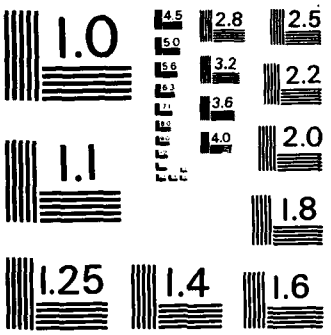
AD-A207 596 VISION-BASED NAVIGATION FOR AUTONOMOUS GROUND VEHICLES 1/2
(U) MARYLAND UNIV COLLEGE PARK CENTER FOR AUTOMATION
RESEARCH L S DAVIS NOV 88 ETL-0519 DACA76-84-C-0004
UNCLASSIFIED F/G 17/5 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

2

ETL-0519

AD-A207 596

Vision-Based Navigation for Autonomous Ground Vehicles Third Annual Report

Larry S. Davis

Center for Automation Research
University of Maryland
College Park, Maryland 20742-3411

November 1988

Prepared for:

Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia 22209-2308

U.S. Army Corps of Engineers
Engineer Topographic Laboratories
Fort Belvoir, Virginia 22060-5546

Destroy this report when no longer needed.
Do not return it to the originator.

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

The citation in this report of trade names of commercially available products does not constitute official endorsement or approval of the use of such products.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) ETL-0519
6a. NAME OF PERFORMING ORGANIZATION University of Maryland 6b. OFFICE SYMBOL (*If applicable*)	7a. NAME OF MONITORING ORGANIZATION U.S. Army Engineer Topographic Laboratories
6c. ADDRESS (*City, State, and ZIP Code*) Center for Automation Research College Park, MD 20742-3411	7b. ADDRESS (*City, State, and ZIP Code*) Fort Belvoir, VA 22060-5546
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Defense Advanced Research Projects Agency 8b. OFFICE SYMBOL (*If applicable*) ISTO	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DACA76-84-C-0004
8c. ADDRESS (*City, State, and ZIP Code*) 1400 Wilson Boulevard Arlington, VA 22209-2308	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 62301E; PROJECT NO.; TASK NO.; WORK UNIT ACCESSION NO.
11. TITLE (*Include Security Classification*) Vision-Based Navigation for Autonomous Ground Vehicles Third Annual Report	
12. PERSONAL AUTHOR(S) Larry S. Davis	
13a. TYPE OF REPORT Annual	13b. TIME COVERED FROM 7/86 TO 7/87
14. DATE OF REPORT (*Year, Month, Day*) November 1988	15. PAGE COUNT 133
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES			18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUB-GROUP	Autonomous Navigation Road Following Computer Vision

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

This is the third annual report for DARPA Sponsored ETL contract DACA76-84-C-0004 (DARPA Order 5096), covering the period July 1986 through July 1987. The report describes both new equipment added to our laboratory and the research performed on autonomous vehicle navigation. We describe the design of a structured light range scanner that has been built and mounted on our robot arm. This scanner provides us with the capability of generating range data similar to that obtainable on the Autonomous Land Vehicle (ALV) using the ERIM scanner. The report also describes the following research projects conducted during the past year:

1) The design and implementation of a rule-based road following system. This system has provided us with a flexible environment in which to experiment with different visual control strategies for road extraction.

2) Road obstacle detection in range data. We have developed computationally simple algorithms for road obstacle detection and applied them to a variety of synthetic and real range imagery. Simple geometric arguments show why these algorithms should be more robust than those currently used on the ALV to detect obstacles. (Continued on next page)

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Linda H. Graff	22b TELEPHONE (*Include Area Code*) (202) 355-2700 22c. OFFICE SYMBOL CEETL-RI

DD Form 1473, JUN 86 *Previous editions are obsolete.*

19. ABSTRACT (Continued)

3) Theoretical analysis of the accuracy of road recovery using motion stereo. Here, our research shows that it is unlikely that the three-dimensional structure of the road can be recovered with sufficient accuracy from motion stereo, given the expected errors in the estimate of vehicle motion.

4) Parallel vision on the Connection Machine. Here, we introduce a computational structure called a Fat Pyramid, and show how the common operation of histogramming can be implemented within the Fat Pyramid structure. Fat Pyramids provide a possible means for effectively utilizing the Connection Machine hardware for either multiresolution or focus of attention vision algorithms.

Finally, the report ends with a discussion of our plans for research during the next three years of our autonomous vehicle navigation research.

Linda H. Graff (202) 355-2700 CEETL-RI

PREFACE

This is the third annual report for DARPA Sponsored ETL Contract DACA76-84-C-0004 (DARPA Order 5096), covering the period July 1986 through July 1987. The report describes both new equipment added to our laboratory and the research performed on autonomous vehicle navigation. We describe the design of a structured light range scanner that has been built and mounted on our robot arm. This scanner provides us with the capability of generating range data similar to that obtainable on the Autonomous Land Vehicle (ALV) using the ERIM scanner. The report also describes the following research projects conducted during the past year:

1) The design and implementation of a rule-based road following system. This system has provided us with a flexible environment in which to experiment with different visual control strategies for road extraction.

2) Road obstacle detection in range data. We have developed computationally simple algorithms for road obstacle detection and applied them to a variety of synthetic and real range imagery. Simple geometric arguments show why these algorithms should be more robust than those used currently on the ALV to detect obstacles.

3) Theoretical analysis of the accuracy of road recovery using motion stereo. Here, our research shows that it is unlikely that the three-dimensional structure of the road can be recovered with sufficient accuracy from motion stereo, given the expected errors in the estimate of vehicle motion.

4) Parallel vision on the Connection Machine. Here, we introduce a computational structure called a Fat Pyramid, and show how the common operation of histogramming can be implemented within the fat pyramid structure. Fat Pyramids provide a possible means for effectively utilizing the Connection Machine hardware for either multiresolution or focus of attention vision algorithms.

Finally, the report ends with a discussion of our plans for research during the next three years of our autonomous vehicle navigation research.

Table of Contents

Index of Figures

1. INTRODUCTION AND SUMMARY

This is the third annual report for DARPA Sponsored ETL Contract DACA76-84-C-0004 (DARPA Order 5096) covering the period July 1986 through July 1987. This report describes the research performed during the final year of our initial three-year contract on autonomous vehicle navigation.

During this year we have made improvements to our laboratory simulation facility—specifically, the construction of a structured light range scanner. This scanner, described in Section 2 of this report, is carried by our robot arm over the terrain board and enables us to simulate the use of the ERIM scanner on the Autonomous Land Vehicle (ALV) in Denver. The range sensor operates by scanning a plane of light over the scene and imaging the scene using a television camera whose position and orientation relative to the plane of light are known. The image points illuminated by the plane of light can have their range recovered using very simple computations.

We have also pursued our program of basic research in visual navigation. Based on our experience with the visual navigation system we designed and implemented during the first two years of the project, we undertook this year to develop a much more flexible and general navigation system based on rule-based and frame systems. Section 3 of this report provides an overview of this system. The system regards the road following task as comprised of two major stages:

1) deciding what objects to look for and where to search for them, and

2) choosing appropriate sensor data analysis techniques for verifying the object's presence and location in the world.

These two functions are performed by the Scene Model Planner and the Scene Model Verifier in the system described in Section 3. The Planner, in addition to interpreting and updating the scene model, initiates queries to a map subsystem based on the vehicle's current navigation goals. The Verifier controls the motion of the sensors and acquires sensor data. In a hypothesize and test paradigm, the Planner sends object hypotheses to the Verifier, while the Verifier returns verified objects (or annotated failures) to the Planner. We have used this system to analyze several short sequences of road images from the Martin test site. The system gives us the ability to rapidly experiment with a variety of strategies for tracking the road through images.

The availability of range data, both from the ALV and from our own range sensor, has stimulated research in range acquisition and range data processing. This research is summarized in Section 4. We describe a set of algorithms for rapidly and robustly detecting obstacles on roads. These algorithms are based on adaptively thresholding directional range differences. For flat roads, it is important for the thresholding to be space variant because the directional range derivatives will vary dramatically across the image. We compare our approach to obstacle detection to two other simple techniques—thresholding height above the ground, and thresholding the difference between observed range and predicted range for a flat road. We show that our technique is much less sensitive to the types of calibration and measurement errors that one would expect to encounter

on the ALV (for example, errors in measuring the tilt of the road with respect to the range scanner).

There are, of course, alternative methods for recovering range. One of these is motion analysis, and we have conducted a series of computational experiments to determine under what conditions one could use analysis of time-varying images to recover sufficiently accurate three dimensional road models. The results, summarized in Section 5, are not very encouraging. It appears that the motion of the ALV through the world must be either known or measurable to an accuracy that we are unlikely to achieve in practice in order to base our road structure recovery on conventional structure from motion algorithms.

During this year the Laboratory took delivery of two Butterfly parallel processors. During the coming year we expect these parallel computers to be augmented by both a Connection Machine and a WARP. Having these machines in the Laboratory has led us to consider certain problems in parallel computing that are relevant to vision and navigation. In Section 6, we discuss some preliminary results on multiresolution image analysis on the Connection Machine. These results are based on a computational model of pyramid processing on the Connection Machine that we refer to as a Fat Pyramid. In a Fat Pyramid, a pixel in the pyramid is represented using a larger and larger number of processors as we move from the base of the pyramid (where a single processor might be used to represent a pixel) towards the apex of the pyramid. Section 6 contains a very fast histogramming algorithm based on Fat Pyramids. We are currently developing fast arithmetic algorithms for the Fat Pyramid and intend to implement a

pyramid image processing system on the Connection Machine based on Fat Pyramids.

Finally, Section 7 contains a list of the reports generated during this last year of the contract.

2. RANGE SCANNER

This section describes hardware, interfaces and algorithms for the CVL light-stripe range scanner. The type of laser ranger used on the ALV is a bulky piece of equipment inappropriate for laboratory use; laser rangers for shorter ranges and suitable for laboratory use on models are under development [Hersman et al.] but were not available at the time of this writing. We found that a *light-stripe range scanner* would be suitable for obtaining range images for model scenes; furthermore, this type of scanner has more points in common with a laser ranger than is apparent at first, and the images obtained are very similar to laser ranger images. This similarity is described in Section 2.1. In particular, dense images can also be obtained; therefore algorithms performing low level processing at the pixel level in the range image prior to 3D reconstruction, such as edge detection, segmentation and edge-preserving smoothing, can be tested on range images obtained by a light-stripe scanner before they are applied to ALV range images.

At this point we need to specify our use of a few terms specific to ranging techniques combining a camera and a projector. These techniques determine ranges by triangulation of the rays of light coming from the projector and rays of light reflected from the patterns created by the projector on the world scene. The generic term for ranging by light-pattern projection is *structured light ranging.* In our specific construction, the projector creates a *light-sheet* or *plane of light.* In all generality, a sheet of light could be a very general ruled surface, but we imply here that the sheet is *flat.* A *sheet* can be allowed a certain thickness;

therefore the term *light-sheet* is useful for discussions about the actual light output of the scanner, and also for considering the actual dispersion of the rays away from an ideal plane, which produces *light-stripes* of nonnegligible thickness when intersecting a world scene. The term *plane of light* will be preferred when referring to the ideally thin planar core of the light-sheet, which illuminates approximately the midpoints of the light stripe. Referring to this planar core is useful in geometric considerations and angular measurements. The method of creating light-stripes on a scene with a light-sheet projector for ranging measurements is sometimes called *light-striping*. We can talk interchangeably of *light-sheet projector* or *light-stripe projector*, the light-stripe being a part and consequence of the light-sheet. But obviously, when considering the camera image, we deal mainly with the *light-stripe*, the only visible part of the light-sheet in clear air. This visual prevalence of the light-stripe made us prefer the term *light-stripe range scanner* to *light-plane range scanner* for designating the whole system.

Range data acquisition using a stripe projector and a camera is a well-known method. For a general overview of this and other range-finding techniques, see [Jarvis]. The projector and the camera are in a known mutual configuration. Ranges of world points illuminated by the light-sheet can be obtained by triangulation, since the distance from the light source to the camera nodal point is known, and both the angles of the sheet of light and the angle of the ray from the illuminated world point to its image can be easily estimated. Variants of this range-measurement method exist and differ in the method of sweeping the scene with the sheet of light:

- In industrial environments, the scene may consist of parts translated on a conveyer belt which have to be recognized. The conveyer belt motion provides the relative sweeping of the parts and the light-sheet.

- For systems mounted on a robot arm, the motion of the arm itself can be used to sweep the stripe [Bolles et al.], [Agin]. In this case, the camera and the stripe projector are displaced together. However, a robot arm is usually not an adequate tool for the small, fast and precise displacements which are required for scanning a light-stripe, because it is a heavy piece of machinery with a lot of inertia, and it makes variable errors according to which joint motions are involved in the displacement of the end plate [Agin].

- A third option has been chosen in our design. A mirror is rotated to project the light sheet at various angles. This type of system incorporates a camera and a light-stripe scanning system that we call a light-stripe range scanner. Light-stripe range scanners are not new. [Shirai] obtained range maps from a light-stripe scanner in 1972 and developed algorithms for recognition of polyhedral objects. [Boyer and Kak] base their preference for a complex color-coded structured-light scheme on the argument that gear lash in the rotating mirror mechanism of a light-stripe range finder can contribute significantly to stripe positioning errors; but we feel that this problem can in fact be minimized to almost any desired degree by using a gearbox with the required (high) precision and by always rotating the gearbox in the same direction.

In our laboratory, the ranger is mounted on the tool plate of the robot arm (Figure 2.1). This does not mean that the robot arm is used for scanning. On the contrary, the robot arm has to be kept still during range image acquisition, while the mirror takes care of the scanning. There are two reasons for mounting the ranger on the robot arm. First, by controlling the robot arm by keyboard input of Cartesian coordinates or angles, it is relatively easy to position the ranger at any desired pose within the operation space of the robot. Second, in a vehicle simulation mode, the displacement of the robot can be driven under software control; the robot motion is then controlled by software which uses

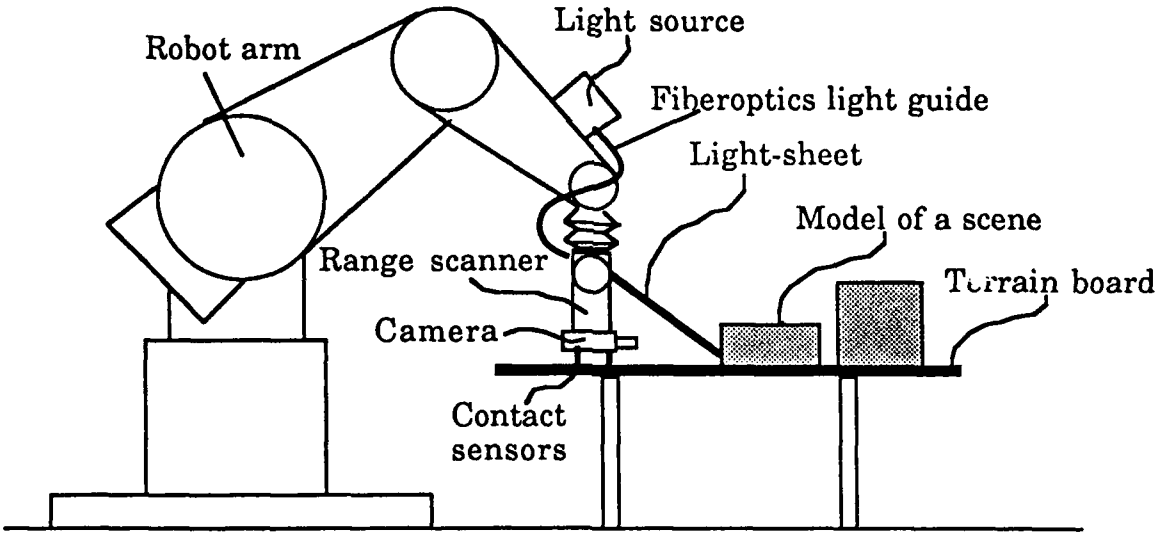


Figure 2.1: Configuration of the Ranging System Mounted on a Robot Arm for Simulation of a Ranger-Equipped Vehicle.

The robot arm itself is not used for light-sheet scanning. This task is performed by a stepper motor rotating a mirror. The robot arm is kept still during range data collection and displaces the range scanner between range images.

range image analysis, in the same way as the trajectory of a vehicle would be controlled by software using laser ranger data [Asada].

One desirable feature which could not be attained is real-time range image acquisition. Our system uses a software stripe search on the digitized camera images and fills each range image pixel with a range value in about 2 milliseconds. This adds up to 8 seconds for a 64×64 image, and close to 8 minutes for a 512×512 image. Hardware solutions are being developed to improve these rates, by application of techniques comparable to those described by [Ozeki et al.].

2.1. Dense Images from a Light-Stripe Scanner

In most of the light-striping techniques described in the literature, the grey levels from the pixels in stripe images from the camera are simply replaced by values coding the ranges to the corresponding world points. The end result after scanning the scene is a striped image of the scene, with the pixel values of the stripes changed to represent the ranges, and the other pixels between the stripes set to 0 (or 1). The problem with such images is that we would like to compress the lines together to eliminate these gaps between stripes, to obtain dense images on which templates can be applied for low-level image-processing operations. If the stripes were straight, they could be pushed one against the other. But the stripes are in fact straight when they are seen from the point of view of the mirror, since a stripe is then viewed within the plane of light which created it. The solution is therefore to obtain the ranges from the mirror, and build an image in

which each row is assigned to the range data obtained for a given stripe. Thus the rows are indexed by the mirror steps. This solution also seems to have been chosen by [Lozano-Perez et al.], for the purpose of doing edge detection on the range image with standard edge operators.

Notice that this is precisely the type of range image also obtained by a laser range scanner doing raster scanning with two mirrors. This parallelism between

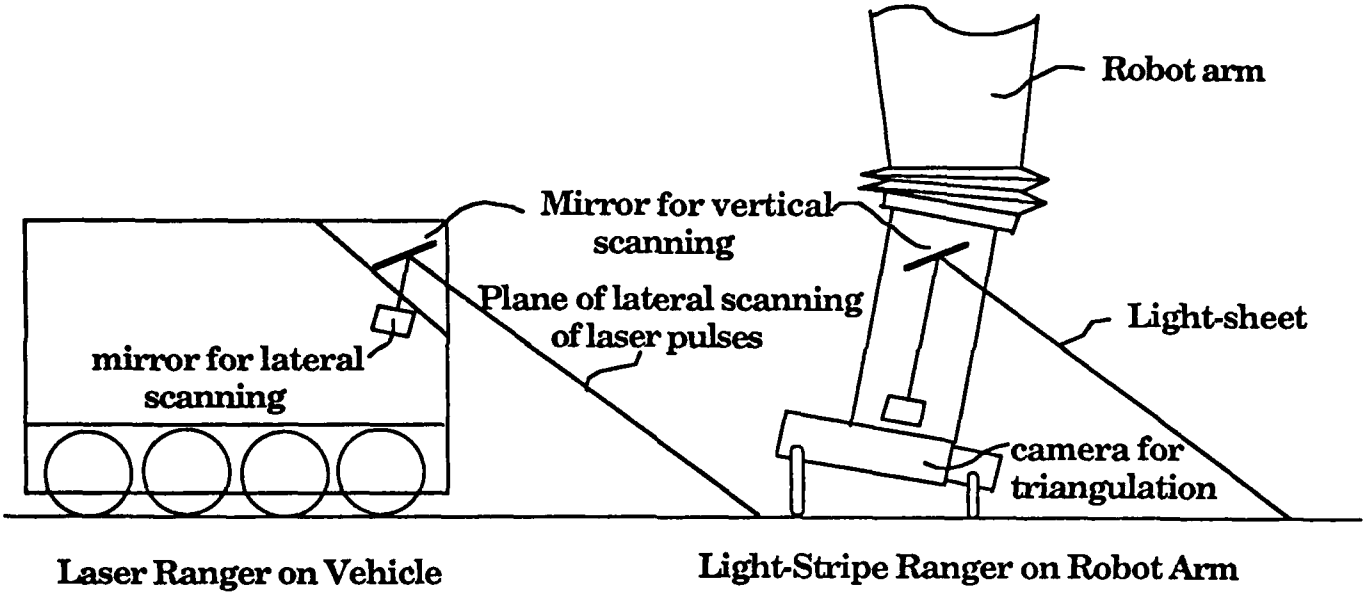


Figure 2.2: Light-Stripe Range Scanner as a Simulator for Laser Rangers.

In a laser ranger, the mirror with the horizontal axis controls vertical scanning. For any given step of this mirror, the pulse is scanned at constant angular increments within a plane which contains the horizontal axis of this mirror, and the ranges for points on the trajectory of the pulse are obtained.
These ranges fill a range image in which a new row is used for each step of the mirror.
Similarly, in the light-stripe ranger, a mirror controls vertical scanning. For any given step of this mirror, the ranges from the mirror to the world points which reflect the light are measured. For any given step of this mirror, these ranges are stored in a specific row of the range image.

the two systems is illustrated in Figure 2.2. For every step of the mirror controlling the vertical scanning, the laser pulse is scanned (by the other mirror) in a plane containing the axis of the first mirror, and sweeps a stripe of world points. The ranges of these world points are placed in the same row of the range image.

2.2. Architecture of the CVL Ranging System

Figure 2.3 illustrates the architecture of the system. This figure shows the elements of hardware, their interconnections, and also the elements of software which control them and the image planes in computer memory where the range images can be found, for display and transfer to other computers.

The following sections of this report are a step-by-step description of the components of Figure 2.3. The order of presentation follows approximately the stream of information from the world points in a scene to the range pixels on the display screen, and the tools which transform this information are presented.

2.3. Description of the Camera-Scanner Block

The frame of the ranger is mounted on the end plate of a robot arm, and supports the camera and the stripe-scanning mechanism (Figures 2.1 and 2.2).

Figure 2.4 provides more details about the structure of the ranger box. The camera is a CCD module producing a video signal. A wide angle lens system with a nominal focal length of 10 mm is mounted on the camera. The parameters of the camera shown in Figure 2.5 were obtained with the lens system focused at 3 ft for an aperture set at f-11. More complete characteristics of the camera are described in [DeMenthon and Asada], along with a description of the

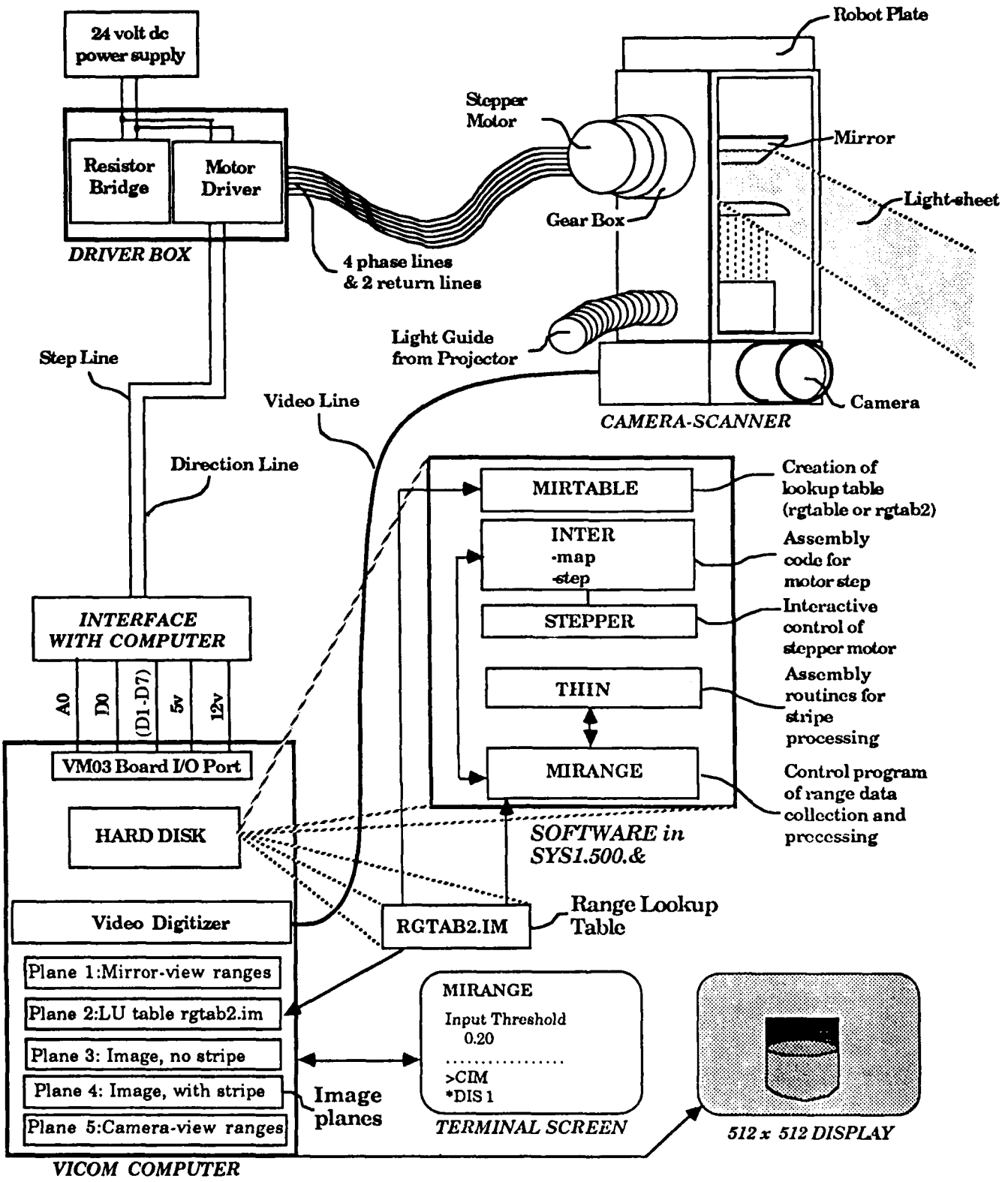


Figure 2.3: Architecture of the Ranging System (Hardware and Software)

The hardware elements are: Camera-Scanner, Driver Box, Computer Interface, Computer, Terminal and Display Screen.
The Software modules are: MIRTABLE, INTER (map and step), STEPPER, THIN and MIRANGE.

calibration method. The video signal is digitized in a VICOM computer, also used to run the software controlling the scanning and range image construction.

The components of the stripe scanning mechanism itself are now described. The 150W light generator itself is not shown in the figures. It is secured away from the robot hand, on a side of the robot forearm. Light is brought from this projector to the ranger box by a 4-ft fiber light guide, which flares to a 2.5" $\times$ 0.125" line of fibers. The line of light from the optic fiber line is further narrowed to 0.1 mm by a brass slit. The diverging light from this slit is refocused to a planar sheet of light by a 4" long cylindrical lens which has a 44 mm focal length. The light-sheet is redirected out of the box of the device by a 3 in. long mirror, a front-silvered microscope slide. The angular displacement of the mirror is precisely controlled by a geared-down stepper motor.

2.4. Optical Performance of the Stripe Projector

The length of a stripe is 200 mm at a distance of 400 mm from the mirror axis. The thickness of the stripe on a plane normal to the light-sheet is 2 mm at 400 mm. The thinnest stripe one could possibly obtain at a given distance is the size of the image of the slit. For the focal length and slit thickness given here, the thickness of this image would be 1 mm at 400 mm, i.e. one-half what we actually obtained. But this result would be achievable only with an axisymmetric lens, instead of cylindrical lenses. The reason is that with the image of a slit through a cylindrical lens, one cannot have good focusing both for rays taking the shortest route from the slit to the lens and for rays travelling at an angle in

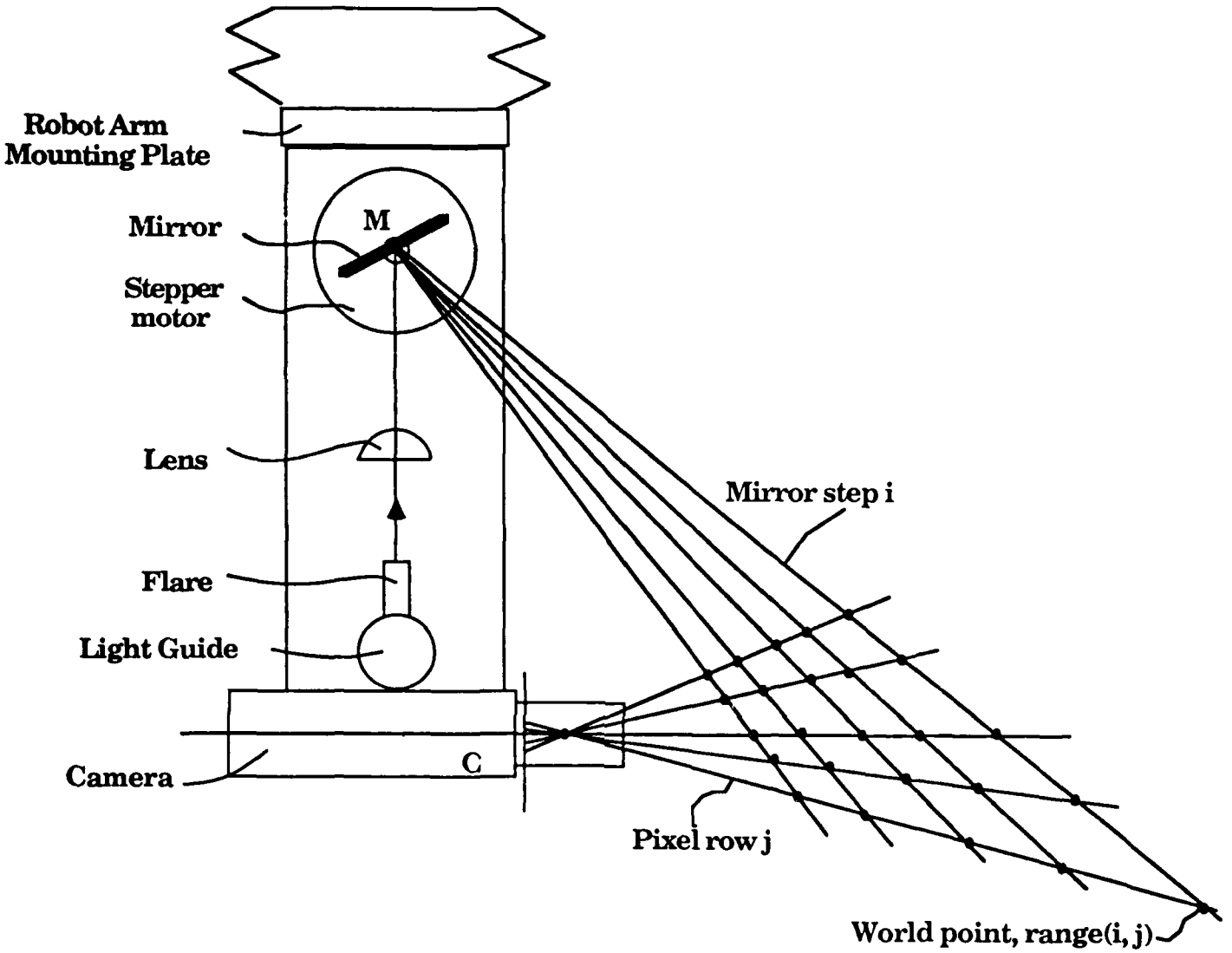


Figure 2.4: Structure of Light-Stripe Range Scanner

The angles of planes of light are quantized by the stepper motor. The camera detection of the vertical angular position of a world point illuminated by a plane of light is quantized by the CCD sensor rows. Thus the measurements of the distances of world points to the mirror axis are subjected to quantization. Precise measurements can only be performed on lines at the intersection of the pencil of planes of light and the pencil of planes of sensor rows (defined by the sensor rows and by the camera nodal point). All points of such lines have the same range, if range is defined as distance to the mirror axis. Thus all measurable ranges can be stored in a 2D lookup table indexed by row numbers and mirror steps. This table maps the physical array of intersections of the two pencils of planes, seen as an array of points in this figure.

the plane defined by the slit and the cylindrical axis of the lens. Unfortunately, an axisymmetric lens would give a 1 mm thick stripe only along the optical axis. Since the slit is very long, the rays from the ends would be subjected to large lens edge distortions.

For lower beam divergence, one would have to resort to a laser source and a cylindrical lens, used this time to spread the narrow beam into a sheet while preserving the thickness of the sheet to the thickness of the original beam. However, the brightness of the stripes would be much lower, since the power in the beam would be only a few mW, whereas the power in our light-sheet is several W.

One should realize, however, that the occurrence of thick stripe images is a generic drawback of light-stripe scanners, regardless of the projection system. Indeed there will always be scene surfaces lit at grazing angles by the light-sheet, producing very thick stripe images.

In any event, in our system only a median curve of the stripe image is used. This median curve is obtained by replacing the two boundary points on the same column of the stripe image by their midpoint. This curve is approximately the image of the part of the stripe created by the median plane of the actual light-sheet. Thus using only the median curve of the stripe is approximately equivalent to working with an ideally thin sheet of light. Errors still occur when stripes have corners, because of the simplification of calculating midpoints column by column. These errors result in a smoothing of the edge of reconstructed objects.

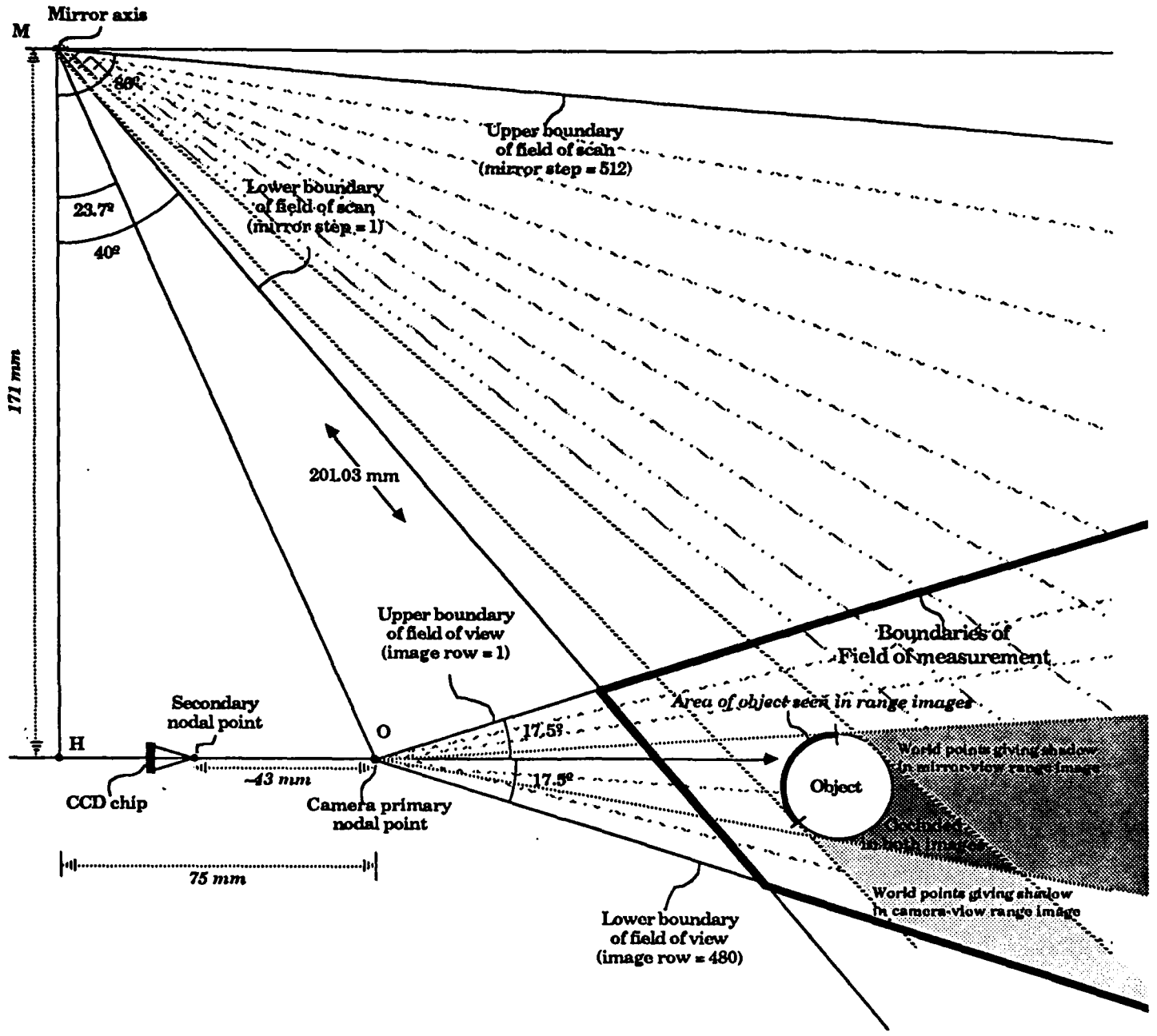


Figure 2.5: Field of Measurements and Shadows

The field of measurement is at the intersection of the camera field of view and the mirror field of scan. Domains of world points giving shadows in the mirror-view range image and in the camera-view range image are shown. Shadows in the mirror-view range image, for example, are defined as images of world points visible from the mirror, but occluded from the camera by the object, and for which the range calculation cannot be performed.

2.5. Motor and Gearbox

The motor rotating the scanning mirror is a stepper motor rated at 5 percent accuracy. This motor is designed for 1.8 degree steps, but it is operated in half-step mode, and mounted on a gearbox with a gear-down ratio of 1:20, giving angular steps for the mirror of 0.045 degrees, and angular displacements of the light-plane of 0.09 degrees. A 360 degree mirror revolution takes 8000 step commands. These specifications were chosen so that in range images of scenes placed a couple of feet in front of the ranger, the camera is able to capture the images of at least 512 distinct stripes, giving 512×512 range images (Figure 2.5).

2.6. Interface with Computer

The stepping motor is controlled by the VICOM computer also in charge of the camera image digitization, by means of the motor driver, and by means of a custom-built interface. This custom interface uses a memory-mapped I/O bus of the VICOM computer. When one writes to specific addresses assigned to a memory-mapped I/O port, the address and data are sent out to this port.

2.7. Home Position of the Mirror

The starting position of the mirror (home position) is taken to be parallel to the optical axis of the camera, so that the initial position of the plane of light is normal to the optical axis.

This position was chosen because it is checked easily with the light projector turned on. In home position, the mirror sends the light back to the slit, and the stripe visible on the brass slit plate is aligned in top of the slit. The mirror

comes back automatically to its home position at the end of the range image production and should be at its home position before starting a new image acquisition.

2.8. Overview of the Range Computation Method

A plane of light is created by the stripe of optic fibers, focused by a cylindrical lens, and rotated to an angle ϕ from its home position by a computer-controlled mirror. It intersects an object in a planar world stripe, and the image of this stripe is seen from the camera and is usually curved (Figure 2.6). This image stripe actually has a thickness of several pixels, and the median curve of this stripe is obtained; for each pixel of this median curve the row in the image is detected. The world point corresponding to this pixel belongs to an image row plane defined by the pixel image row and by the camera image center. The concurrent knowledge of the angle of this image row plane and the angle of the light-plane allows a calculation of the specific distance from the world point to the *mirror axis* by triangulation. In practice these calculations are performed only once for all the ranges to the mirror axis of all the intersections between the pencil of image row planes and the pencil of light-planes within the camera field of view and the mirror field of scan (Figure 2.4). The results of these calculations are placed in a two-dimensional lookup table. The next section gives more details on the construction of this table and the calculation of the ranges.

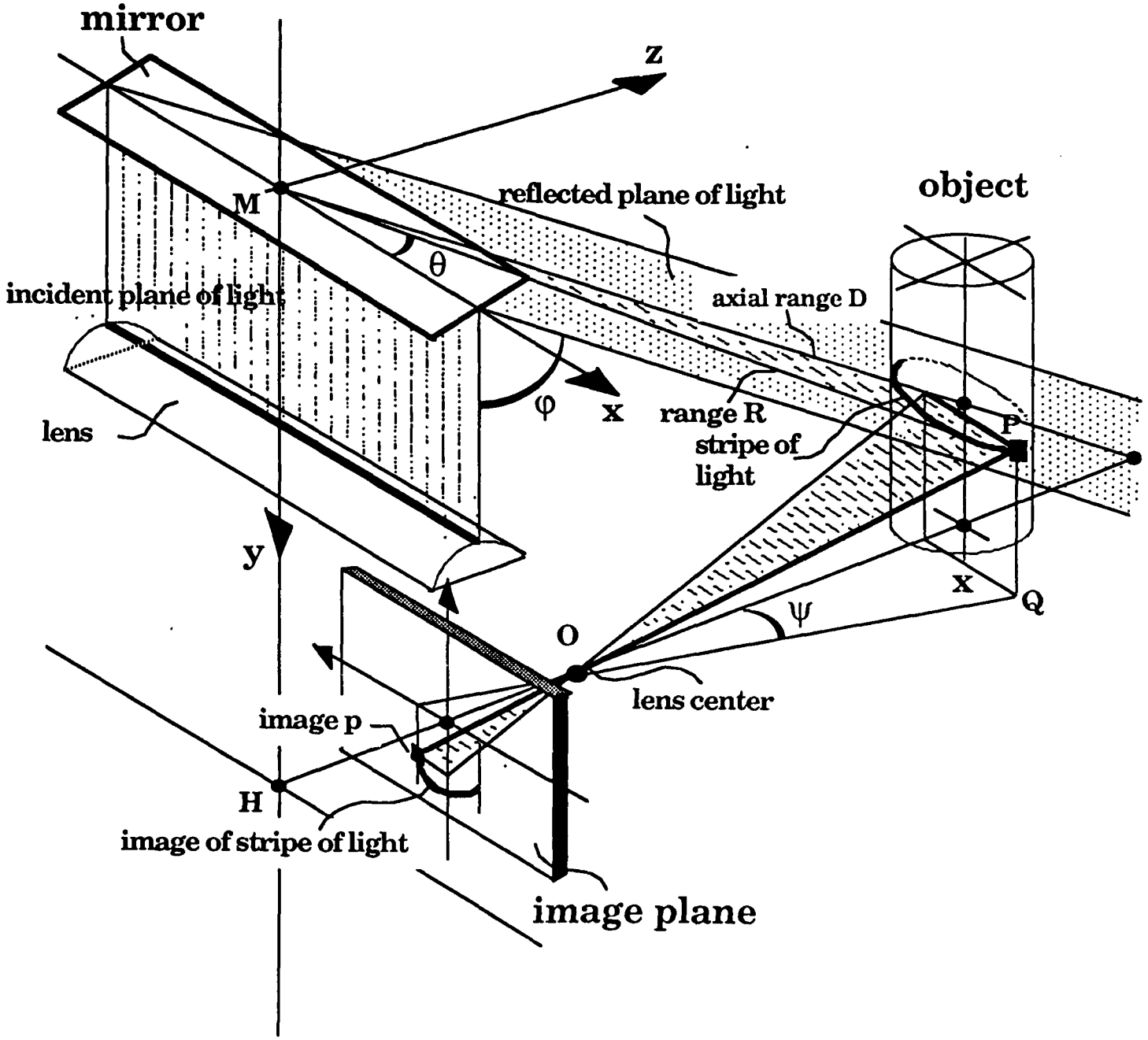


Figure 2.6: Perspective View of Geometric Relations between World Scene, Mirror Angle, Plane of Light and Image of Stripe on Image Plane.

In a mirror-view range image, the column numbers of the pixels are the original columns of the images p of world points P, while the row numbers are the mirror step numbers, related to ϕ, the gray levels represent the range D. From the column number, one can deduce Ψ, and with ϕ and HO, calculate the Cartesian coordinates of P.

2.9. Justification for Axial Ranges and a 2D Range Lookup Table

As we just mentioned, the range lookup table lists ranges from the mirror axis instead of ranges from the center of the mirror. The reason is that all the measurable axial ranges can be stored in a 2D table, whereas a 3D table would be required if ranges from a point were used. Indeed all the world points which have their images on the same image row when illuminated by the same plane of light have the same axial range, independently of their image columns. Thus the table need only be indexed by the mirror steps and the image rows, not the camera image columns. To see this, consider that world points which have their images on the same row when they are illuminated by the same light-plane are at the intersection of the light-plane with the image row plane defined by the image row and the lens center (Figure 2.7). This image row plane is parallel to the mirror axis because the mirror axis was made parallel to the image rows, thus the intersection between this plane and the light-plane is also parallel to this axis. Therefore the world points of this intersection have the same axial range. Since there is only a limited number of image rows and mirror steps, all the possible axial ranges can be precalculated and stored as elements of a 2D array indexed by the mirror steps and image rows.

2.10. Calculation of Axial Ranges

Now details are given for the calculation of the axial range D of a world point P on a stripe created when the plane of light is generated at step i of the mirror. This is the calculation performed to fill up the range lookup table. It

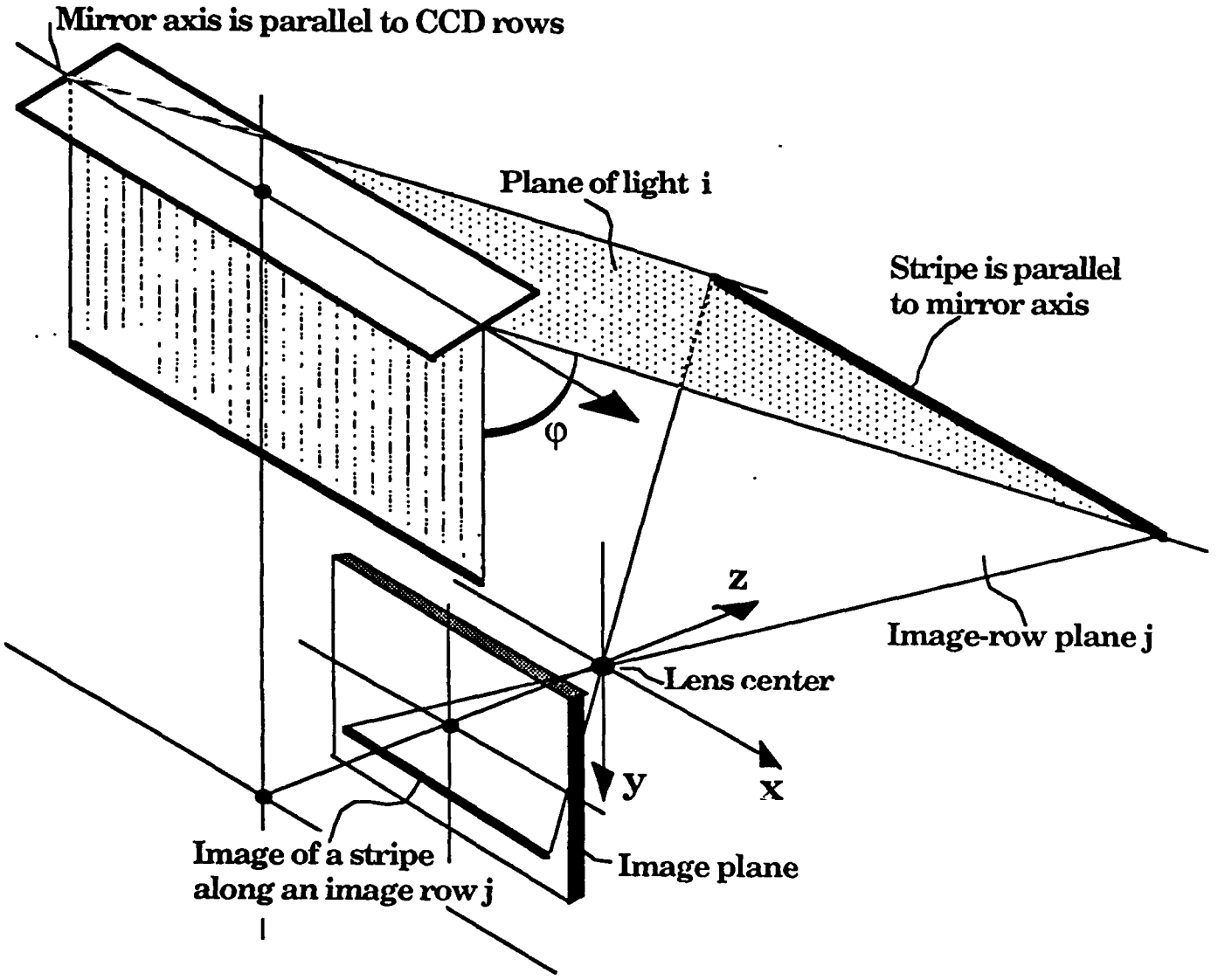


Figure 2.7: Justification for the Construction of a 2D Lookup Table for Ranges Defined as Distances to the Mirror Axis.

For a given mirror step i, image points belonging to the same image row j correspond to world points which have the same range with respect to the mirror axis. Indeed, such world points belong to a line parallel to the mirror axis. This is due to the fact that the mirror axis is set parallel to the camera sensor rows. Thus a plane containing a row is parallel to the mirror axis, and the world points are at the intersection of such a plane and the plane of light. Therefore, one just needs to build a 2D range table indexed by the mirror steps i and the image rows j. The ranges in this table are distances to the mirror axis and not distances to a point.

assumes that the specific row of the image of the point P has been given, by an image processing routine described in the next section. Figure 2.8 shows the relative positions of P, the mirror and the camera, in an orthogonal projection in the direction of the mirror axis. In such a projection, distances to the mirror axis are conserved, thus the calculation of range with respect to the mirror axis can be done in the plane of the figure. O is the lens center, M the projected location of the mirror axis. The range calculation applies the classic *law of sines* in the triangle *MOP*. The length OM is a parameter defined by the structure of the system, and the angles α and β are defined respectively from the row number of the image of P and from the mirror angle.

$$D = \frac{OM \sin\alpha}{\sin(\alpha+\beta)}$$

α is expressed in terms of known parameters:

$$\alpha = \frac{\pi}{2}+\gamma+\delta$$

with

$$\delta = \text{Arctan}\left[\frac{row-C_y}{f_y}\right]$$

in which

row is the row number of the image of P in screen pixels,

f_y is the focal length in the y-direction in screen pixels,

C_y is the row number of the image center in screen pixels,

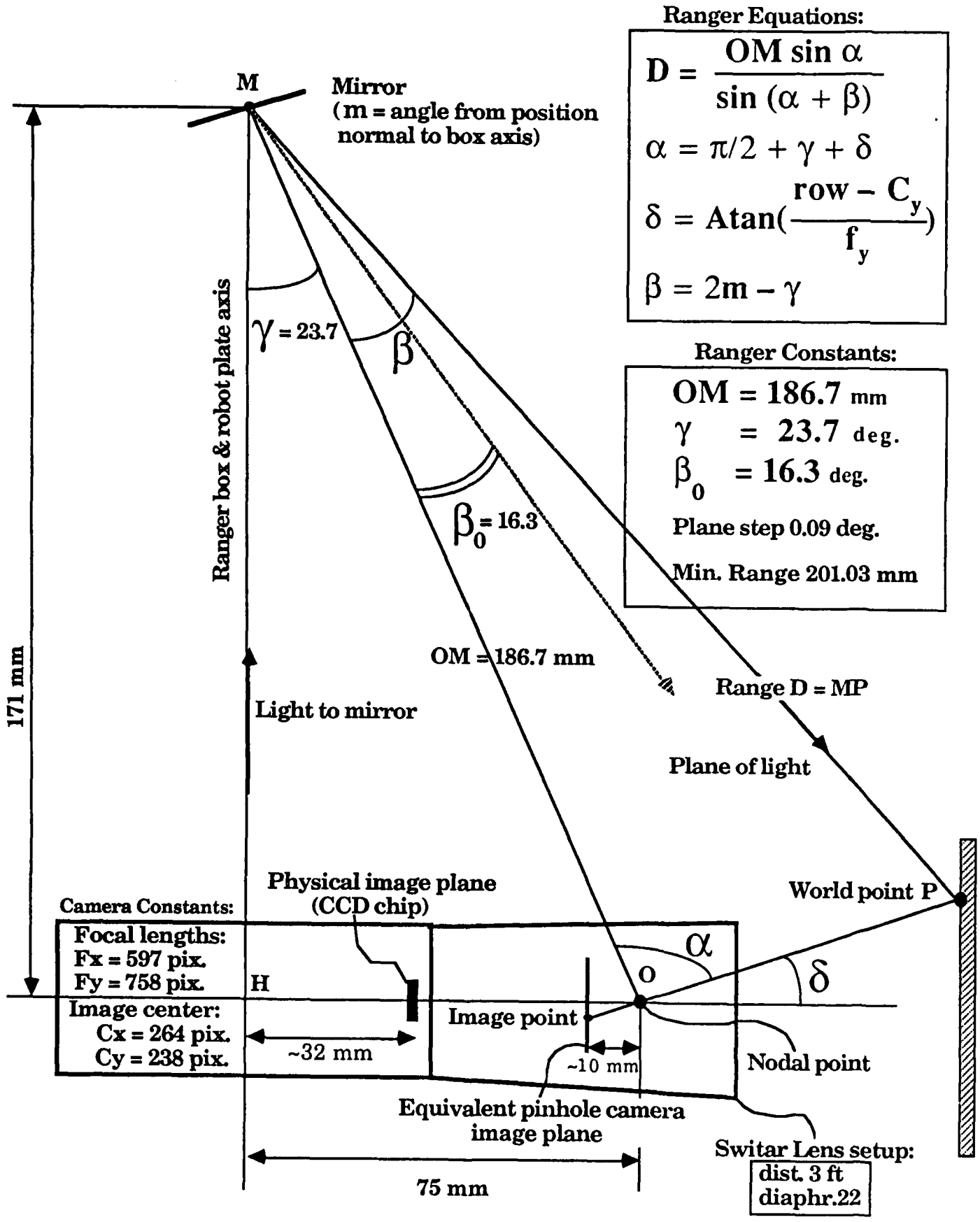


Figure 2.8: Ranger Calibration Parameters and Variables

β is now expressed in term of known parameters:

$$\beta = 2m - \gamma$$

where γ is the angle which the line OM makes with the starting position of the plane of light, m is the angle given to the mirror from its home position to the position which gave the plane of light illuminating the world point P.

$$m = stepangle\,(n_o + i_1 - 1)$$

Here, *stepangle* is the angle of one mirror step. The total number of mirror steps from home, i, has been decomposed into two numbers, n_o and i_1. n_o is the number of steps which bring the mirror to its initial operating position, and i_1 is the number of mirror steps from this initial operating position.

In our design, *stepangle* = 0.045 degrees. The initial operating position is for a mirror angle of 20 degrees and a light-plane angle of 40 degrees from home position. This corresponds to an angle β (denoted β_0 in Figure 2.8) of 16.3 degrees for the light-plane. The number of steps n_o is 444. The values of OM, γ, C_y and f_y obtained by the calibration procedure described in [DeMenthon and Asada] are listed in Figure 2.8.

2.11. THIN Routine

Every time the mirror is stepped and creates a new stripe on the world scene, the camera image which contains the image of the new stripe is grabbed and its address is sent to the routine called THIN. This routine scans the image column by column, from the position of the previous stripe and going up in the

image, to locate in each column the lower and upper bounds of the stripe (Figure 2.9). A lower bound of the stripe is detected when the difference between the no-stripe image pixel and the stripe image pixel is above a specified threshold. At the upper bound of the stripe this test fails again. The pixel row of the midpoint between these bounds is determined, and its range is read in the range lookup table. The lookup table is indexed by the pixel position in the image column and by the mirror step count. Finally, the pixel with its grey level coded by range is added to the range image.

This algorithm assumes that increasing mirror steps move images of the stripe up in the camera image. In other words, the bottom edge of the current stripe is assumed to be above the bottom edge of the stripe processed at the previous step. The mirror is rotated with respect to the scene in such a direction that this is verified in most of the cases. This hypothesis greatly reduces the time of search for stripes in the picture. However, in the cases of objects flying above the ground, it could happen that a sheet of light *above* the object would make a stripe on the ground which is visible *under* the object, in the gap between the ground and the object. Thus this strategy would give larger unexplored areas (shadows) on the ground than a more complete search for stripes.

To limit the number of false detections of stripe pixels, a thickness of at least two pixels is required for a stripe. Similarly, a drop in brightness is considered a top edge of the stripe if this drop is maintained for at least two succeeding pixels in the scanned column.

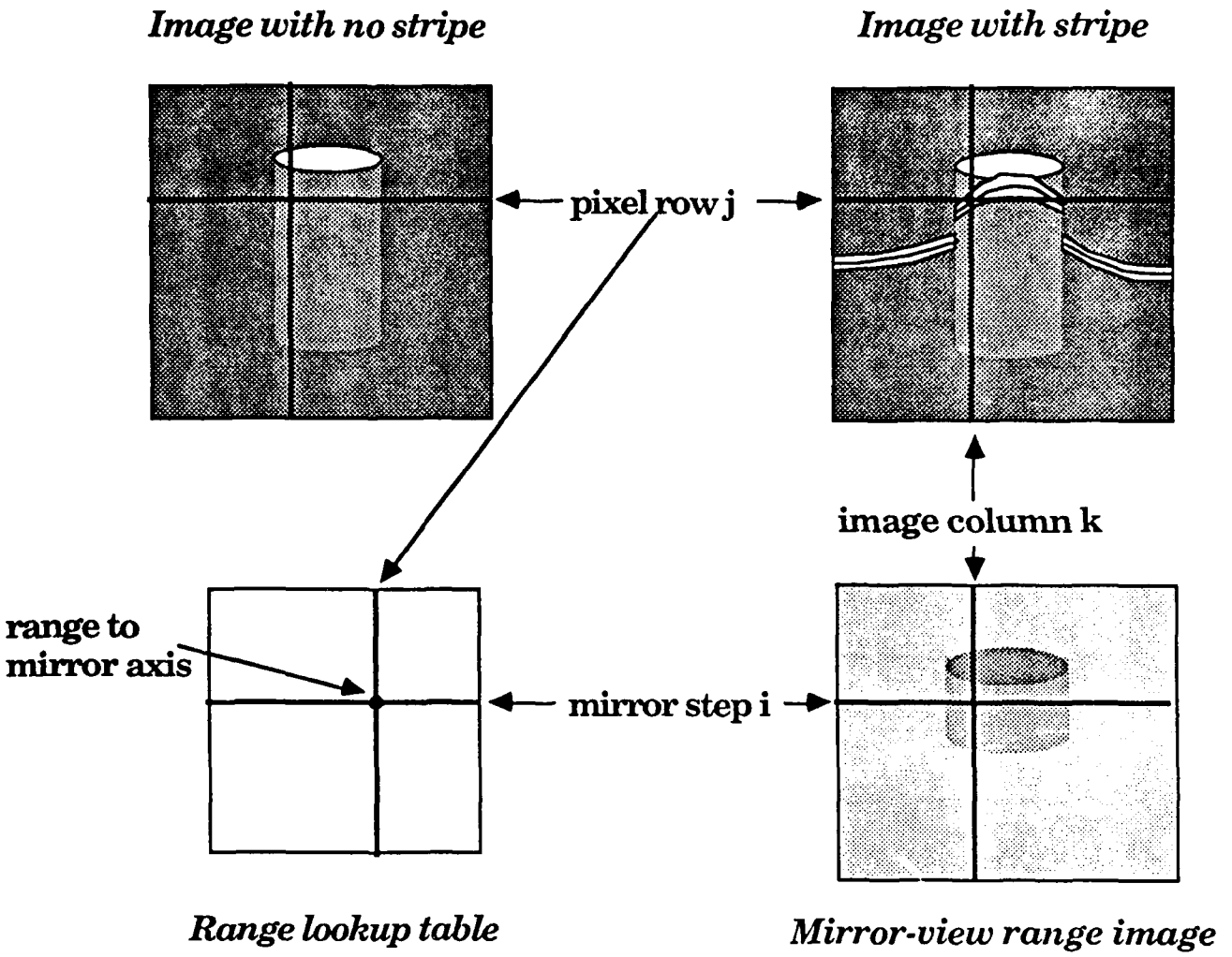


Figure 2.9: Creation of a Mirror-View Range Image from Images of Light Stripes.

The image with stripe and the image without stripe are simultaneously scanned upward column by column. A pixel at the bottom of the stripe is detected when the difference between the grey levels of the two images is above a given threshold. Then the scan is extended until the difference falls back under the threshold, which gives the top pixel of the stripe. The row of the midpoint is deduced, and used along with the mirror step count to read in a lookup table the range of the corresponding world point with respect to the mirror axis. This range is coded and placed in the mirror view range image at the same column position as the original image pixel and at a row equal to the mirror step count.

2.12. Conclusions

We have described the CVL implementation of a light-stripe range scanner. This scanner can produce images from models of complex natural scenes, for developing algorithms from low level processing to navigation. These solutions could be immediately applicable to robotic vehicles equipped with the small inexpensive laser rangers to come.

References

[1] P. Veatch and L.S. Davis, "IRS: A Simulator for Autonomous Land Vehicle Navigation", University of Maryland, Center for Automation Research Technical Report 310, July 1987.

[2] M. Hersman, F. Goodwin, S. Kenyon and A. Slotwinski, "Coherent Laser Radar Application to 3D Vision", Vision '87, Detroit, June 8–11, 1987.

[3] R.A. Jarvis, "A Perspective on Range Finding Techniques for Computer Vision", *IEEE-PAMI*, vol. PAMI-5, pp. 122–139, 1983.

[4] B.C. Bolles, J.H. Kremers and R.A. Cain, "A Simple Sensor to Gather Three-Dimensional Data", SRI, Technical Note 249, July 1981.

[5] G.J. Agin, "Calibration and Use of a Light Stripe Range Sensor Mounted on the Hand of a Robot Arm", Carnegie-Mellon University Robotics Institute Technical Report 85-20, November 1985.

[6] Y. Shirai, "Recognition of Polyhedrons with a Range Finder", *Pattern Recognition*, vol. 4, pp. 243–250, 1972.

[7] K. Boyer and A.C. Kak, "Color-Encoded Structured Light for Rapid Active Ranging", *IEEE-PAMI*, vol. PAMI-9, no. 1, January 1987.

[8] M. Asada, "Building a 3D World Model for a Mobile Robot from Sensory Data", University of Maryland, Center for Automation Research Technical Report 332, October 1987.

[9] O. Ozeki, T. Nakano and S. Yamamoto, "Real-Time Range Measurement Device for Three-Dimensional Object Recognition", *IEEE-PAMI*, vol. PAMI-8, no. 4, July 1986.

[10] T. Lozano-Perez, J.L. Jones, E. Mazer, P.A. O'Donnell and W.E. Grimson, "Handey: A Robot System that Recognizes, Plans, and Manipulates", 1987 Int. Conf. on Robotics and Automation, vol. 2, pp. 843–849, March 1987.

[11] D.F. DeMenthon and Minoru Asada, "Calibration of a Light-Stripe Range Scanner", University of Maryland, Center for Automation Research Technical Report (in preparation).

3. A RULE-BASED SYSTEM FOR VISUAL NAVIGATION

The DARPA Autonomous Land Vehicle (ALV) Program involves the development of computer vision techniques by which a vehicle can autonomously navigate itself through the environment. Although the goals for the ALV are broad, including both on- and off-road navigation, our work has primarily been concerned with the road following task. Figure 3.1 presents a view as seen from a camera mounted on top of the ALV at the Martin Marietta test site. From images such as these, the ALV vision system constructs a model of the environment; this scene model contains the objects visually identified by the ALV. Based on this collection of objects, the vehicle plans a course and moves through the environment.

For the road following task, the scene model contains either objects that represent the road or objects from which the location of the road can be deduced.

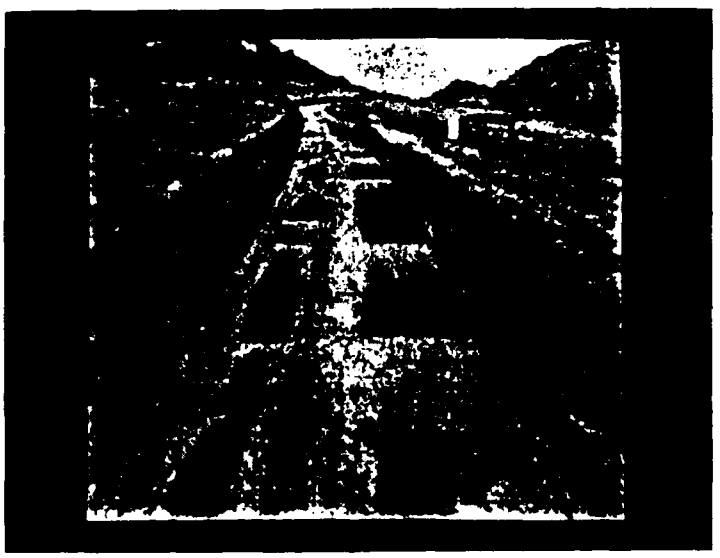


Figure 3.1: A Typical ALV Road Image

Obviously, the direct detection of a patch of road would be most useful; however, in the event that the ALV vision system cannot directly identify the road, the detection of other objects may suggest the location of the road. For example, telephone poles and ditches often run parallel to the road; their presence may thus provide clues as to its location and direction. In certain cases, major landmarks contained in a road map such as buildings may be used to infer the road location; however, such information is more useful in registering the vehicle to some absolute location.

Faced with the task of building a scene model, the ALV vision system must decide which of the above objects should be sought in order to locate the road. Hence, the first step in constructing the scene model is the joint decision of what object to search for in the world and where to search for it. The second step performed by the ALV vision system is to gather evidence confirming the existence of the specified object at the hypothesized location. The third and final step involves reasoning about the evidence. If the confirming evidence is sufficient, the object is inserted into the scene model; otherwise, the object may be hypothesized elsewhere or a different object hypothesized.

Construction of the scene model is complex. The selection of which object to track depends on the navigation goals of the ALV, the history of object tracking, the contents of the scene model, and information from the road map. Verifying the existence of an object requires directing vehicle sensors towards the object, fusing data from different sensors, and selecting algorithms for image analysis. Methods for performing all these tasks are continually evolving as the

road following task becomes better understood. New objects must be tracked by the ALV, new sensors are available to track objects, and new image processing techniques are identified for sensor image feature extraction. The successful evolution of an ALV vision system hinges on the ability of its control structure and knowledge representation schemes to accommodate these changes.

We shall describe the design of a system for constructing an ALV scene model, offering a flexible control structure able to accommodate new strategies for object tracking, sensor selection, and feature extraction. The goal of the system is to provide a flexible tool for the development of ALV road following software. The design is based on concepts described in [Hanson and Riseman], but offers a unique implementation based on a set of communicating production systems.

The scene model is constructed as a network of frames, each frame corresponding to a class of objects and encapsulating the relevant information pertaining to that class. The control structure used to build the network is based on a system of communicating production systems implementing a structured blackboard. Each region of the blackboard corresponds to a particular class of frames and contains the rules which define the attributes of the class. The system promotes modularity and maintainability through a structured object representation and a structured control scheme.

In the following section, we provide an overview of the system before exploring in detail the representation and control schemes. In Section 3.2 we discuss object modeling and describe the object classes currently available in the system, while in Section 3.4 we discuss the scene model network. Sections 3.4 and 3.5

discuss the control strategies involved in the selection and verification of objects, respectively. We conclude with a series of results demonstrating the system's capabilities.

3.1. System Overview

The task of building a scene model for the ALV consists of two major subtasks:

1. deciding what object to look for and where to look for it;
2. verifying that the object exists in the world.

These two functions are performed by the Scene Model Planner (the Planner) and Scene Model Verifier (the Verifier), respectively; together, they form the Scene Model Builder (the Builder). The data flow diagram for the Builder is presented in Figure 3.2. The Planner, in addition to interpreting and updating the scene model, is aware of the local navigation task and initiates queries to the a priori road map. (The local navigation task is a function of the global navigation task and the location of the vehicle; for example, a local navigation task might be to follow the road for 100 meters, or turn right at the first intersection after a certain landmark is identified. The road map contains a priori information about the ALV environment, including the approximate locations, sizes, and compositions of roads, intersections, and landmarks.) The Verifier controls the movement of the sensors and acquires the sensor image data. In a hypothesize-and-test paradigm, the Planner sends object hypotheses to the Verifier, while the Verifier returns verified objects to the Planner.

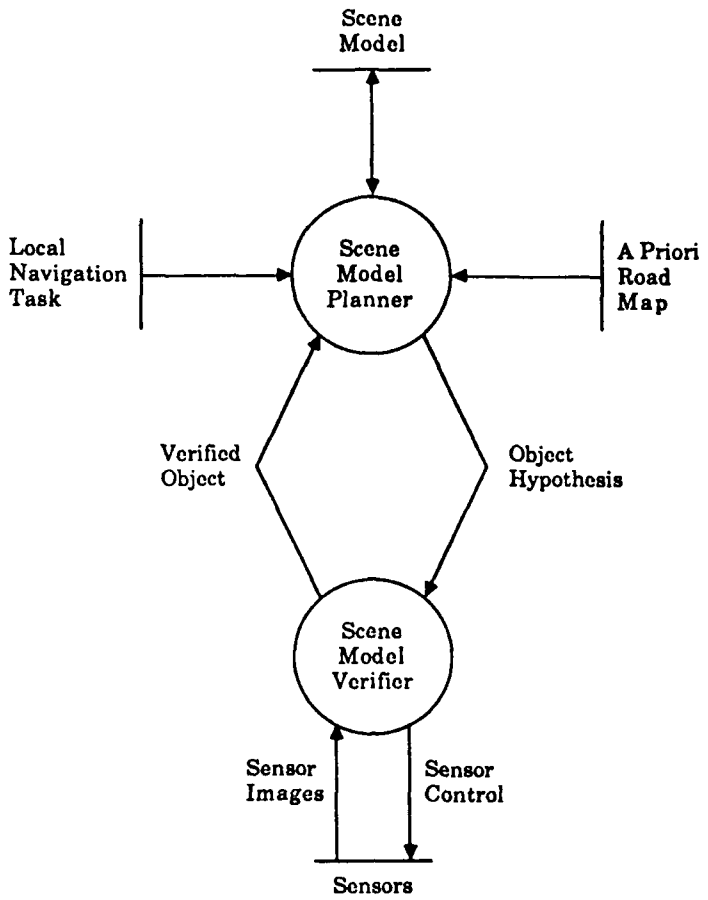


Figure 3.2: The Scene Model Builder Dataflow

The dataflow of the Builder proceeds as follows. The Planner first determines the scene model requirements of the local navigation task; for example, following a straight road requires that the left and right road boundaries be contained in the scene model. Next, the Planner looks at the road map and the partial scene model and decides what objects may be useful in locating the road; for example, it might decide that a road patch, a ditch, or even a row of telephone poles is sufficient to define a road boundary. The Planner then decides the type and expected location of the object to be tracked, hypothesizes the object, and passes the hypothesis to the Verifier.

The Verifier attempts to verify the hypothesis by directing the vehicle's image sensors towards the expected location of the object. The object is then located in the sensor images and its image location is mapped to a 3-D location based on a fixed point of reference. The confidence with which the object is found becomes a measure of its verifiability. Once the confidence is defined, the hypothesis is returned to the Planner for inspection. If the object is deemed sufficiently verified, it is added to the scene model. Otherwise, the Planner determines the next course of action; for example, the object may be hypothesized in a different location, or a new object hypothesized.

3.2. Modeling World Objects

Objects in the ALV scene model exhibit the following relationships:

- *Component Relationships.* For example, an intersection is made up of four connecting roads, stop lights, etc. These component objects, in turn, can be decomposed into their component objects, e.g. a road may be defined as a pair of left and right segments, each edge representing the border between road and shoulder, or shoulder and background. Primitive objects such as lines and surfaces extracted from an image cannot be decomposed.
- *Spatial Relationships.* For example, telephone poles are often located near the road and run parallel to the road.
- *Property Inheritance.* For example, a three-dimensional line segment may be defined by a pair of endpoints. The edge separating the road surface from the shoulder is a specialization of a three-dimensional line segment; thus, in

addition to its properties specific to a road edge, it inherits the endpoint properties of the three-dimensional line segment.

To accommodate these relationships, frames have been chosen to model objects [Minsky]. A frame is a data structure containing a set of slots (or attributes) which encapsulate the relevant knowledge pertaining to an object. Slots may contain values, e.g. the width of a road patch, or pointers to related frames, e.g. component and spatially related frames. Property inheritance among frames is accomplished by supplementing a frame's slots with the inherited frame's slots.

3.2.1. The Road Patch

A planar ribbon is defined as a pair of facing and parallel three-dimensional line segments. A road patch is a specialization of a planar ribbon, whose three-dimensional line segments represent the left and right features of the road. Thus,

planar-ribbon
attributes:
search-location
search-strategy
back-connected-planar-ribbon
front-connected-planar-ribbon
has-part-left-world-segment
has-part-right-world-segment
expected-width
actual-width
actual-width-confidence
parallelism-confidence
total-confidence

road-patch
attributes:
prior-road-straightness
left-world-segment-type
right-world-segment-type
inherited-frames:
planar-ribbon

Figure 3.3: The Road Patch/Planar Ribbon Frames

a road patch frame inherits the attributes of a planar ribbon. The road patch and planar ribbon frames are depicted in Figure 3.3. Road patches are oriented and may be connected together to form a piece of road; the front of one road patch may be connected to the back of another. The orientation of a road patch is based on the assigned orientation of the initial road patch; typically, the back end of the initial road patch is closest to the ALV, while the front end is furthest from the ALV. The left and right features, i.e. three-dimensional segments, are oriented looking from the back to the front of the road patch. Figure 3.4 depicts the vehicle with respect to a series of road patches.

3.2.2. The Road Patch Segment

A world segment is defined as a three-dimensional line segment. A road patch segment is a specialization of a world segment representing a road feature, i.e. the boundary between the road surface and the shoulder surface or the boundary between the shoulder surface and the vegetation or background. Thus a road patch segment frame inherits the attributes of a world segment. The road patch segment and world segment frames are depicted in Figure 3.5. Road patch segments are oriented and may be connected together to form a continuous linear feature; the front of one road patch segment may be connected to the back of another. The orientation of a road patch segment is based on the orientation of its parent road patch.

A camera segment is defined as a two-dimensional line segment extracted from a camera image. A road patch camera segment is a specialization of a cam-

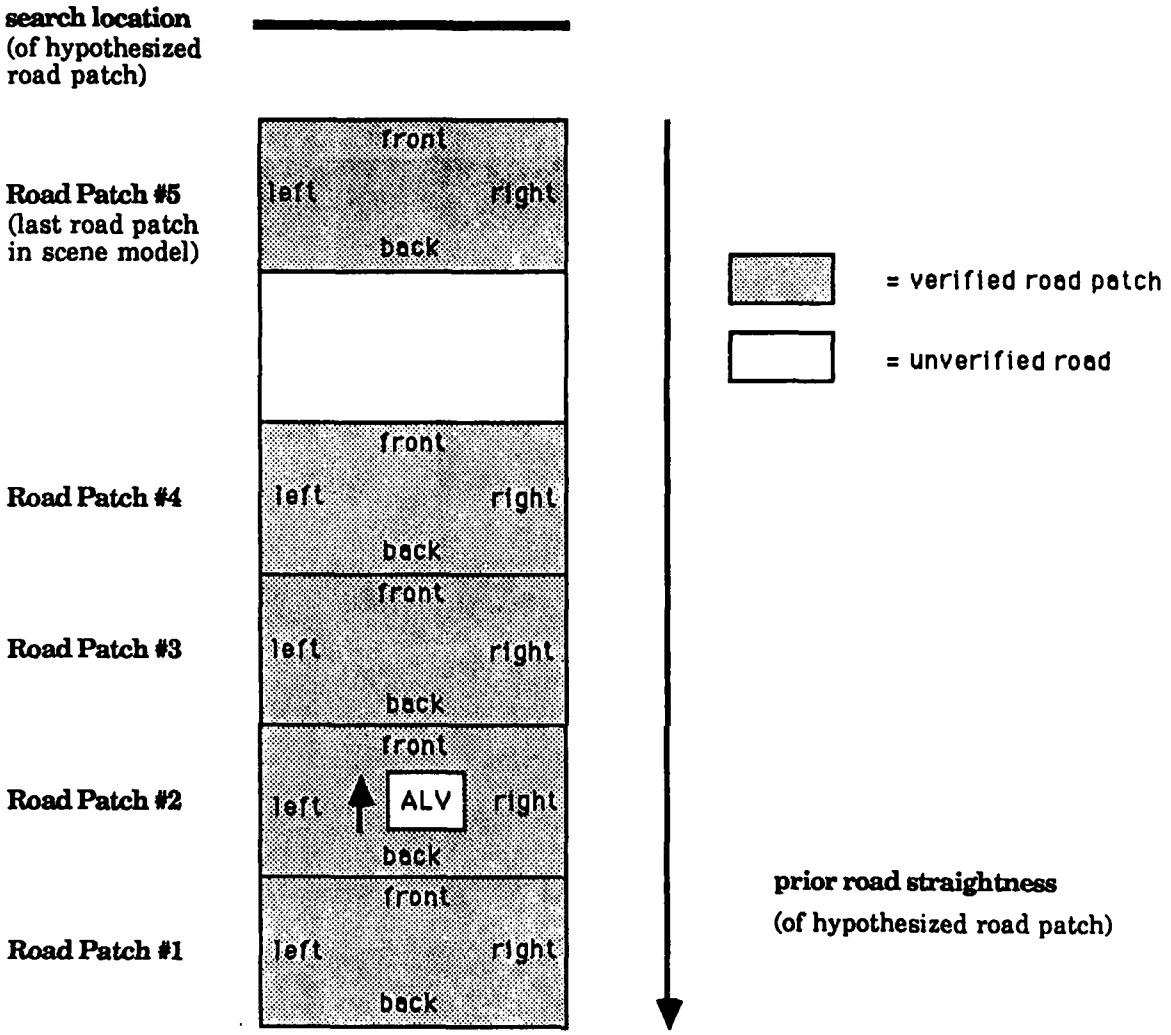


Figure 3.4: A Series of Road Patches

era segment representing the two-dimensional projection of a three-dimensional road feature. Thus a road patch camera segment frame inherits the attributes of a camera segment frame. The road patch camera segment and camera segment frames are depicted in Figure 3.6. Road patch camera segments are oriented and

world-segment
attributes:
search-location
search-strategy
endpoint-A
endpoint-B
endpoint-A-connected-world-segment
endpoint-B-connected-world-segment
part-of-planar-ribbon
has-part-camera-segment
total-confidence

road-patch-segment
attributes:
world-segment-feature
endpoint-A-orientation
endpoint-B-orientation
continuity-confidence
inherited-frames:
world-segment

Figure 3.5: The Road Patch Segment/World Segment Frames

camera-segment
attributes:
search-window
search-strategy
endpoint-A
endpoint-B
endpoint-A-connected-camera-segment
endpoint-B-connected-camera-segment
part-of-world-segment
camera-image
method-of-extraction
total-confidence

road-patch-camera-segment
attributes:
camera-segment-feature
endpoint-A-orientation
endpoint-B-orientation
inherited-frames:
camera-segment

Figure 3.6: The Road Patch Camera Segment/Camera Segment Frames

may be connected together to form a continuous two-dimensional linear feature; the front of one road patch camera segment may be connected to the back of another. The orientation of a road patch camera segment is based on the orientation of its parent road patch segment.

3.3. The Scene Model

The scene model is central to the ALV vision system. It is accessed by the Planner in determining what object to search for and by the ALV navigator when plotting a course through the scene model objects. As the domain of scene model objects grows larger, so does the variety of requests to the scene model. It is important that the semantics of the access commands be stable while the structure of the scene model evolves to accommodate new object classes. Hence, the structure of the scene model is transparent to those modules that access it. As in the case of the scene model objects, the scene model is a frame defining a set of query functions and hiding all implementation details. For example, queries can be made to determine the length of the straight road at the end of the scene model, or what road boundary, i.e. road or shoulder edge, has been verified most successfully.

Given the present limited domain of scene model objects, i.e. road patches and their components, a connected graph of road patches serves as an adequate scene model. Figure 3.7 depicts a scene model containing three road patches; two of the road patches are connected to each other while the third is disconnected. Note that the links between a road patch and its component road patch segments, and between a road patch segment and its component road patch camera segment, are shown. In addition, the network includes the links between connected road patches, road patch segments, and road patch camera segments.

The frames comprising the road patches in Figure 3.7 can be divided into two layers. The upper layer, including the road patch frame and its two road

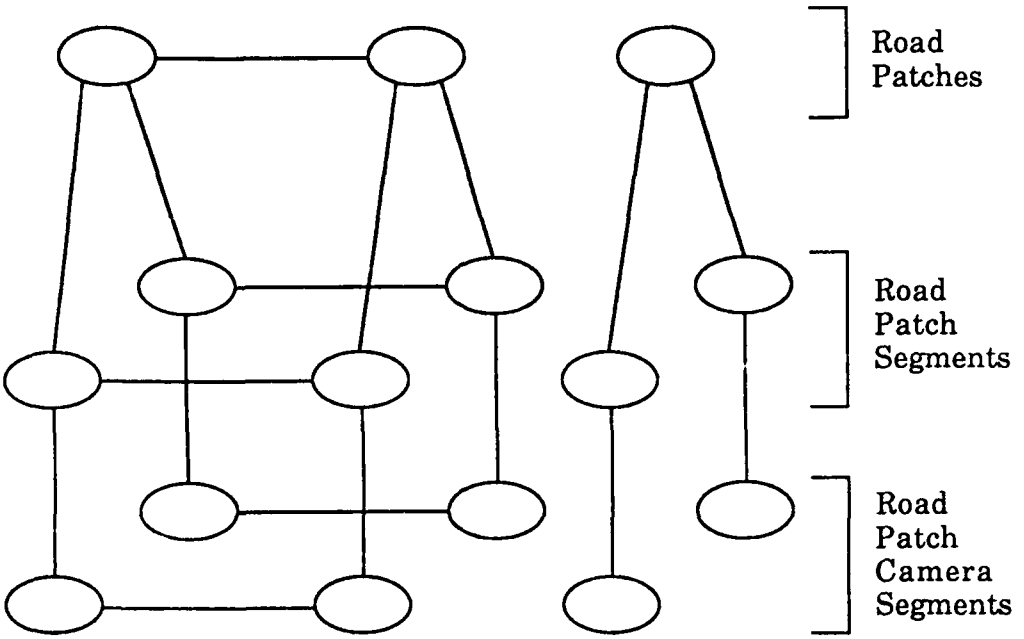


Figure 3.7: The Scene Model Network

patch segment component frames, contains objects and their components as they are defined in the world. The lower layer, including the road patch camera segment frames, contains objects as they are defined in the sensor images. This division facilitates the incorporation of new sensors to the vehicle. For example, if we add a range scanner to the ALV, then a road patch segment frame would be given an additional slot defining a component road patch range segment. Aside from the additional attribute, the only impact on the road patch segment frame would be an alteration of the *endpoint A (B)* calculation to include the data from the road patch range segment, and an alteration of the total confidence function to include the road patch range segment total confidence.

3.4. The Scene Model Planner

The function of the ALV navigator is to examine the scene model and plot a course for the vehicle with respect to the objects. If the local navigation task is to navigate the vehicle down the center of the road, then in order for the navigator to be successful, the scene model must contain a sufficient number of relevant objects. For example, a scene model containing a series of connected road patches would be sufficient; the navigator need only plot a smooth curve between the left and right components of each road patch. However, a scene model containing houses would not provide sufficient information for the navigator to plot a path down the road; the navigator would be unsure of the proximity of the houses to the road. It is up to the Planner to decide, based on the local navigation task, what objects should appear in the scene model. However, the Planner's decision to track a particular object depends on more than the relevance of the object to the local navigation task. Choosing an object for verification should also depend on the history of tracking that object. For example, if the Planner repeatedly fails to verify a particular class of object, it should hesitate to attempt further verification; however, if verification fails due to the object being far from the vehicle, this should not prevent the Planner from attempting to verify the object if the vehicle moves closer to the object.

The Planner is implemented as a frame whose slots point to the modules with which the Planner communicates, i.e. the scene model, the a priori road map, the local navigation task, and the verifier. The unique aspect of this frame is that it inherits the capabilities of a production system, providing a rule

database, a factual database, and a conflict resolution strategy. Based on the local navigation task, the a priori map, and the scene model, the production system decides what type of object to track and verify. To simplify the initial implementation of the Planner, we have assumed a constant local navigation task of following the road ad infinitum, and an a priori road map which contains the approximate locations of intersecting roads along with an approximate road width. Hence, the production system consistently selects a road patch for verification by instantiating a road patch frame whose attributes are undefined.

The next task of the production system is to choose the search location of the road patch hypothesis. The production system first queries the scene model for the directional history of the road. If the direction has varied erratically, then the Planner's confidence as to the location of the road patch is low. Imposing the constraint that the hypothesized road patch must be connected to the last road patch in the scene model improves the likelihood of verifying the road patch hypothesis. Hence, the production system defines the search strategy as "connected"; the search location is defined to be the leading edge of the connected road patch. If the road has been found to be recently straight for, say, at least 10 meters, then the Planner assumes that the road beyond the scene model is also straight. In this case the search strategy is defined to be "disconnected" and the search location is extrapolated a distance of 10 meters from the end of the scene model. If successful, the scene model can be built more rapidly in this fashion, ultimately resulting in higher vehicle speeds.

Before the road patch hypothesis is transmitted to the Verifier, the Planner must decide which left and right features should define the road patch. This decision is based on the features defining the nearest verified road patch and the confidence with which those features were verified. The expected road width is defined as the actual road width of the nearest verified road patch in the scene model.

The Planner is now ready to send the road patch hypothesis to the Verifier, where evidence is gathered in support of it. Once complete, the Verifier returns the hypothesis to the Planner; all the attributes in the road patch frame are now defined. If the evidence is deemed acceptable by the Planner, it will add the verified object to the scene model. However, if the evidence is considered unacceptable, several options are available to the Planner:

1. hypothesize the object at a different location;
2. hypothesize a different object;
3. retain the verified components of the unverified object.

Currently, only option 1 is implemented and proceeds as follows. If the unverified road patch hypothesis is disconnected, i.e. the Planner ventured out beyond the end of the scene model to hypothesize the road patch, the hypothesis is abandoned and a connected road patch is hypothesized back at the end of the scene model. If the unverified road patch hypothesis is connected, the Planner aborts the road following task. Figure 3.8 summarizes the actions taken by the Planner. When additional object classes become defined, the Planner may take

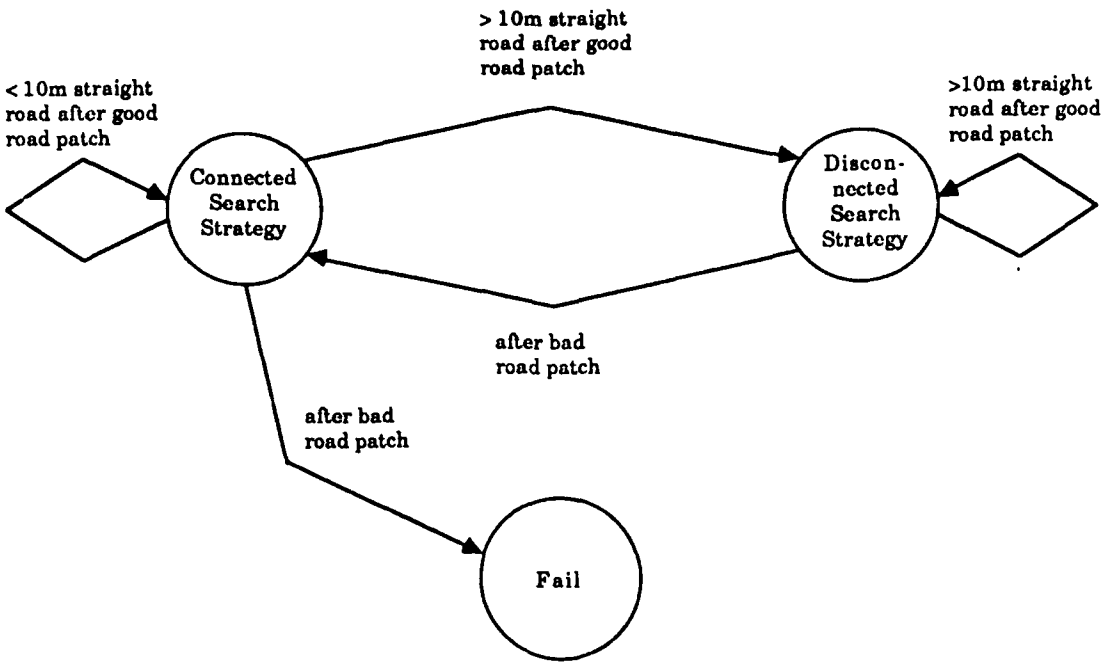


Figure 3.8: The Scene Model Planner States

advantage of option 2; for example, the Planner may choose to hypothesize a ditch beside the road if it was unsuccessful in verifying the road. When different search strategies become defined, the Planner may take advantage of option 3. For example, although a road patch may not have been successfully verified, one of its component road patch segments may yield a high confidence; the Planner may choose to insert this component into the scene model.

3.5. The Scene Model Verifier

The role of the Verifier is to receive an object hypothesis from the Planner, collect evidence in support of the object, and return the verified object to the Planner. More specifically, when the Verifier receives an object hypothesis in the form of a sparsely defined frame, it proceeds to fill in the empty attributes; if the

object has component parts, e.g. a road patch segment, the Verifier must create and define these frames. To accomplish this task, a separate blackboard has been assigned to each class of object. In Section 3.5.1 we describe the structure of an object blackboard; the mechanism by which blackboards communicate to verify an object is presented in Section 3.5.2.

3.5.1. The Object Blackboard

When an object hypothesis is posted on the blackboard corresponding to its class, knowledge sources are activated to fill the empty attributes of the hypothesis. As in the case of the Planner, each blackboard is implemented as a frame, providing a set of attributes and inheriting the capabilities of a production system. The attributes provide links to other blackboard frames and system modules, e.g. vehicle pilot and image processor. The production system rules control the activation of knowledge sources, i.e. when the left-hand side of a rule matches the contents of the factual database, the right-hand side activates a knowledge source. Ties are resolved by the conflict resolution strategy.

3.5.2. Top Down Hypothesis Verification

When the Planner hypothesizes an object, the Verifier invokes a top-down approach to verify the object. In Figure 3.9, this approach has been applied to the verification of a road patch hypothesis. Once the Planner creates the road patch hypothesis, it posts it on the road patch blackboard (a specialization of a planar ribbon blackboard). The rules belonging to the road patch blackboard, acting as daemons, invoke knowledge sources to define the attributes of the now

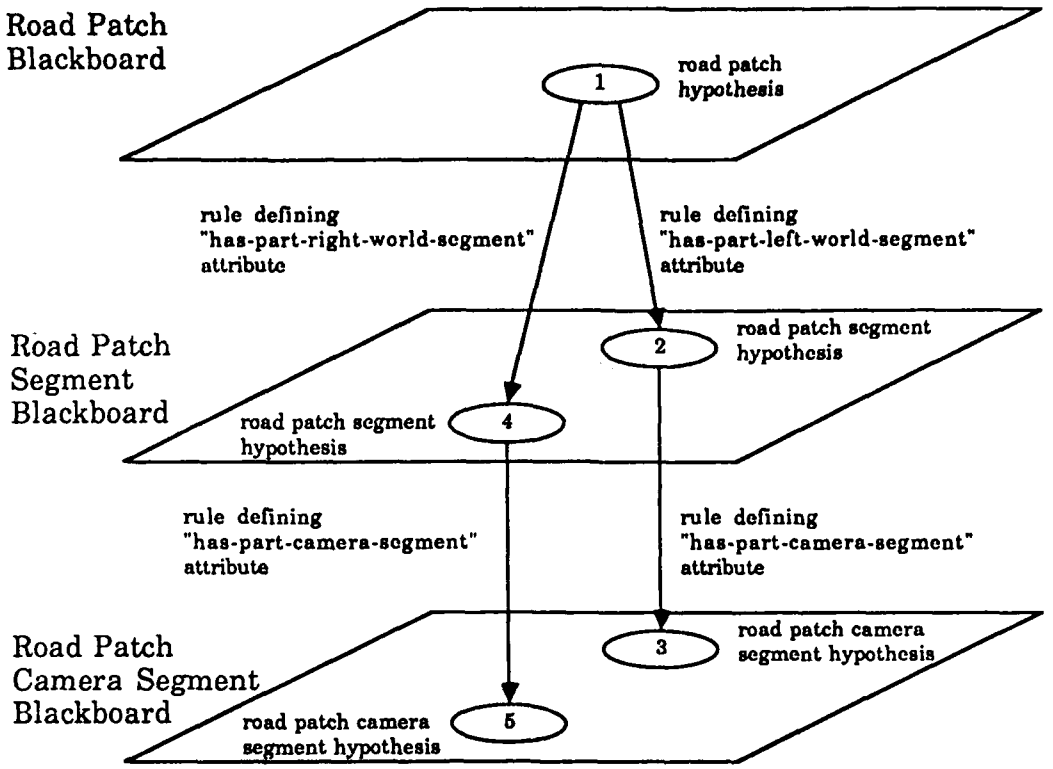


Figure 3.9: Top-Down Hypothesis Verification

sparsely defined road patch hypothesis. The rule antecedents ensure that the attributes are defined in a specific order. When rules fire to define the *has part left world segment* component, the activated knowledge source creates a road patch segment object hypothesis, defines a subset of its attributes, and posts it on the road patch segment blackboard. At this point, control is transferred to the road patch segment blackboard while the road patch blackboard is put to sleep.

Responding to a new object hypothesis on their blackboard, the rules belonging to the road patch segment blackboard proceed to define the attributes of the road patch segment object. When rules fire to define the *has part camera seg-*

ment attribute, the activated knowledge source creates a road patch camera segment, defines a subset of its attributes, and posts it on the road patch segment blackboard. Control is transferred to the road patch camera segment blackboard and the road patch segment blackboard is put to sleep.

The rules at the road patch camera segment blackboard proceed to define where and how the road patch camera segment is to be searched for in the camera image. When the rules fire to define *endpoint A (B)*, a knowledge source applies the *method of extraction* to the *search window contained in the camera image*. The results define the *endpoint A (B)* and *total confidence* attributes. At this point, all the attributes are defined and control is passed back up to the road patch segment blackboard where the remaining road patch segment hypothesis attributes are defined. Similarly, when its attributes are defined, control is passed up to the road patch blackboard. The next attribute, the *has part right world segment*, repeats the entire process until eventually control is once again at the road patch blackboard. Once the last attribute, the road patch *total confidence*, is defined, the completed road patch hypothesis is returned to the Planner for evaluation.

3.6. Experimental Results

In this section we demonstrate the system on two road sequences; the road images were taken from the Martin Marietta ALV test track in Denver, CO. The current implementation runs in the University of Maryland Franz Lisp environ-

ment [Allen et al.], under UNIX[1] 4.3BSD on a VAX[2] 11/785. As described earlier, all system modules are frames implemented using the Maryland Franz Flavors package [Wood]; the production system frames inherited by the Planner and Verifier blackboards are implemented using YAPS [Allen]. YAPS is an antecedent-driven production system similar to OPS5 [Forgy], but offering more flexibility. Functions bound to the frames are implemented in Lisp; C routines are called from the Lisp environment for numerically-intensive processing. All image display functions are provided by a Vicom image processor.

The first sequence is shown in Figure 3.10 and demonstrates the construction of a scene model containing a straight road. At the bottom of the image, the initial search windows are placed according to the a priori points on the side of the road; the search windows are indicated by the rectangular boxes which contain

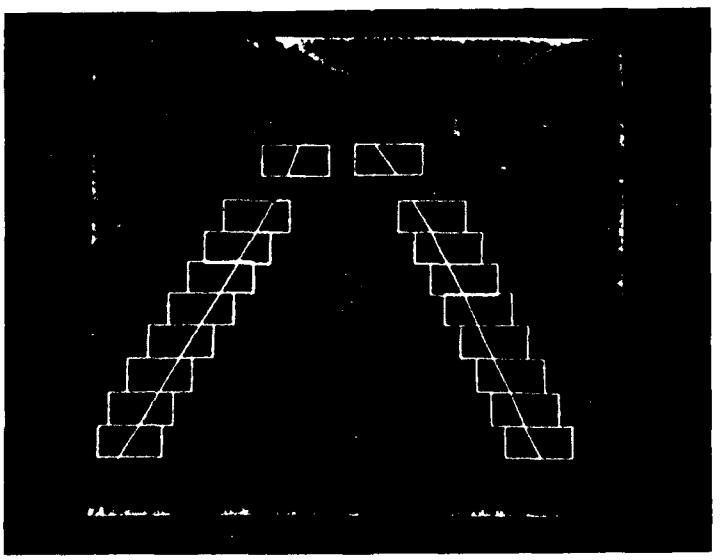


Figure 3.10: Tracking a Straight Road – Frame 1

[1]UNIX is a trademark of Bell Laboratories.

[2]VAX is a trademark of Digital Equipment Corporation.

the extracted line segments. From then on, the road patch connected search strategy is repeatedly invoked to verify successive connected road patches. Following the insertion of the eighth road patch into the scene model, over 10 meters of straight road have been accumulated. In this case, the disconnected search strategy is invoked resulting in a search location 10 meters beyond the end of the scene model. As the road was correctly predicted to be straight, the road patch is successfully verified. With approximately 20 meters of straight road in the scene model, the disconnected search strategy is again invoked; however, when the three-dimensional search location of the third road patch is mapped to the current image, the search windows are out of bounds (off the top of the image). Subsequently, as depicted in Figure 3.11, a new image is acquired and the windows mapped to the correct locations; again, the road patch is successfully verified.

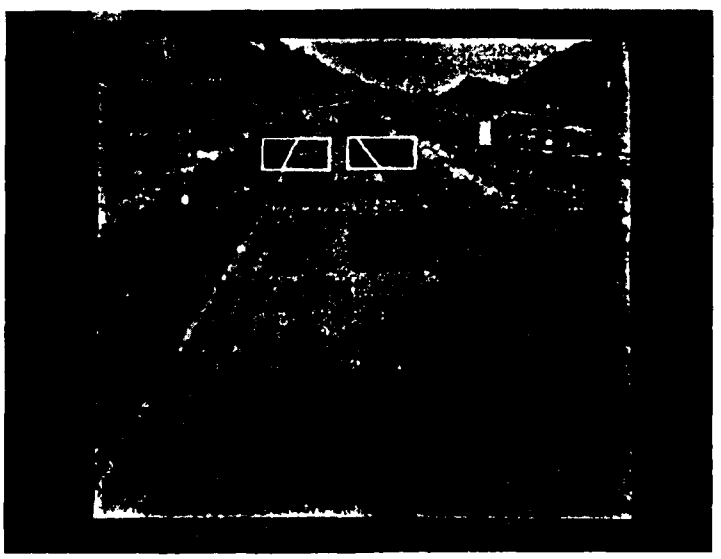


Figure 3.11: Tracking a Straight Road – Frame 2

In the second sequence, shown in Figure 3.12, the ALV attempts to track a curved road. As in the previous sequence, the initial portion of the road is straight; the same steps are used to build the initial eight road patches and the search strategy is changed from connected to disconnected. However, because the vehicle could not predict the upcoming curve in the road, the predicted search location is off the road, ultimately yielding a road patch with very poor total confidence (due to lack of parallelism and poor width). The Planner aborts the disconnected search strategy and invokes the connected search strategy from the previously verified road patch in the scene model. As a result, the curve is successfully navigated, as shown in Figure 3.13.

3.7. Related Work

The decomposition of an object both by component and by level of abstraction, and the construction of hierarchical frame networks, bear close resemblance

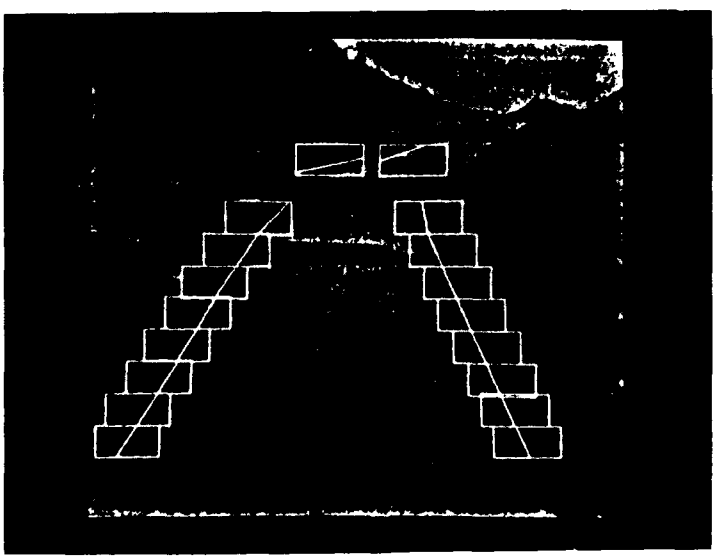


Figure 3.12: Tracking a Curved Road – Frame 1

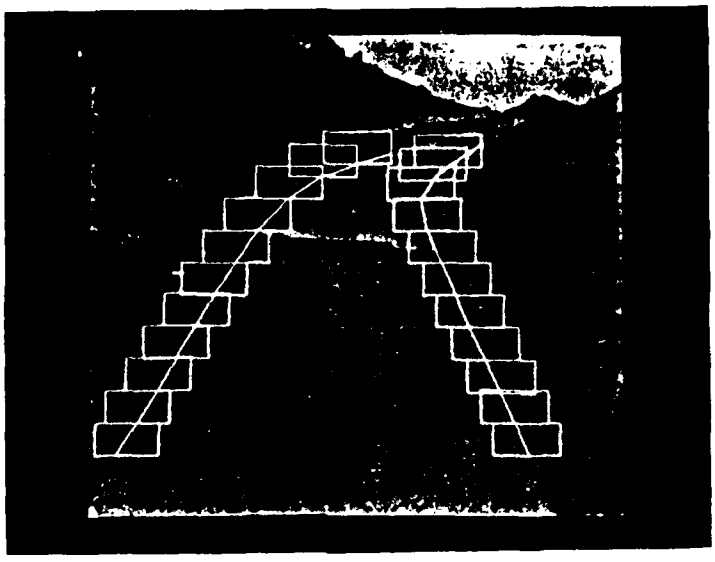


Figure 3.13: Tracking a Curved Road – Frame 2

to techniques used in the VISIONS system [Hanson and Riseman]. In the VISIONS system, the long term memory (LTM) contains a priori visual knowledge of the world, while the short term memory (STM) represents the interpretation of the scene. Both the LTM and STM are structured as a hierarchy of levels of representation defining the levels of object abstraction. The control strategy first decides which partial model (frame network) to focus on, expands (hypothesizes) a node, and finally verifies the node. Although originally defined for outdoor house scenes, this work has recently been extended to the road following task [Arkin et al.].

[Lawton et al.] describe a system also resembling the VISIONS system. The short term memory (STM) acts as a dynamic scratchpad for the vision system, containing object hypotheses, incoming imagery, and the results of feature extraction. When hypotheses accumulate sufficient evidence, they are moved to the

long term memory (LTM), which includes a priori terrain representations. The control structure provides both top-down and bottom-up hypothesis instantiation over the network hierarchies.

Although hypothesis instantiation in the above systems is both top-down and bottom-up, the entire sensor image is processed to initialize the short term memory with image features; local image processing in our system is based on a Hough transform approach [LeMoigne et al.]. In addition, none of these systems use a rule-based production system to invoke knowledge sources.

[Smith and Strat] describe an information manager that is the core of a sensor-based autonomous system. A centralized knowledge database is proposed, accessible to a community of independent asynchronous processes. The representation scheme organizes data tokens in both an *octree* and a semantic network thus supporting both spatial and semantic queries. The independent asynchronous processes can be activated by daemons embedded in the database or by procedure call.

3.8. Conclusions

The system described in this section provides a flexible architecture for constructing an ALV scene model. The representation of objects as networks of frames offers a natural grouping of knowledge; the multiple layers of abstraction facilitate the addition of new sensor features in support of existing world objects. Construction of the frame network is provided by a set of modular blackboards providing top-down instantiation of the frames in the network. Each blackboard,

implemented by a production system, is an "expert" in defining a particular class of frame; the English-like rules governing the invocation of knowledge sources are easy to understand, and narrow the gap between control specification and implementation. From a system maintenance standpoint, all object frames, blackboards, production system tools, and object oriented programming tools are off-the-shelf; these facilities are documented, tested, and readily accessible. The implementation languages supporting the system cover the needs of the programmer; YAPS offers high-level encoding of control strategy, Lisp provides symbolic manipulation, C speeds up numerical processing, and Flavors facilitates inter-object communication.

The system is currently being expanded to support new planning and verification strategies. The Planner is being supplemented with strategies for road following in the event that a connected road patch cannot be verified. This includes proceeding past an unverified road patch provided that it contains a verified component, and invoking an exhaustive search for road patches in a given area; the latter strategy will be accomplished using bottom-up verification in which road patch camera segments posted at lower levels generate instances of road patches at upper levels. This integration of top-down and bottom-up verification will remove some of the burden placed on the Planner of accurately predicting the location of an object.

References

[1] E.M. Allen, R.H. Trigg and R.J. Wood, "The Maryland Artificial Intelligence Group Franz Lisp Environment, Variation 2", Technical Report TR-1226, Department of Computer Science, University of Maryland, November 1983.

[2] E.M. Allen, "YAPS: Yet Another Production System", Technical Report TR-1146, Department of Computer Science, University of Maryland, December 1983.

[3] R.C. Arkin, E.M. Riseman and A.R. Hanson, "AuRA: An Architecture for Vision-Based Robot Navigation", Proceedings: 1987 DARPA Image Understanding Workshop, pp. 417–431, Los Angeles, CA, February 1987.

[4] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, pp. 393–396, John Wiley & Sons, Inc., 1973.

[5] C.L. Forgy, "OPS5 User's Manual", Technical Report CMU-CS-81-135, Department of Computer Science, Carnegie-Mellon University, July 1981.

[6] A.R. Hanson and E.M. Riseman, "VISIONS: A Computer System for Interpreting Scenes", in A.R. Hanson and E.M. Riseman, (eds.), *Computer Vision Systems*, Academic Press, Inc., 1978.

[7] D.T. Lawton, T.S. Levitt, C.C. McConnell, P.C. Nelson and J. Glicksman, "Environmental Modeling and Recognition for an Autonomous Land Vehicle", Proceedings: 1987 DARPA Image Understanding Workshop, pp. 107–121, Los Angeles, CA, February 1987.

[8] J. LeMoigne, A.M. Waxman, B. Srinivasan and M. Pietikäinen, "Image Processing for Visual Navigation of Roadways", Technical Report CAR-TR-138, Center for Automation Research, University of Maryland, July 1985.

[9] M. Minsky, "A Framework for Representing Knowledge", in P.H. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill, Inc., 1975.

[10] G.B. Smith, and T.M. Strat, "Information Management in a Sensor-based Autonomous System", Proceedings: 1987 DARPA Image Understanding Workshop, pp. 170–177, Los Angeles, CA, February 1987.

[11] B. Srinivasan, "Image Processing Algorithms for Navigating Roadways", MS Thesis, University of Maryland, 1984.

[12] R.J. Wood, "Franz Flavors: An Implementation of Abstract Data Types in an Applicative Language", Technical Report TR-1174, Department of Computer Science, University of Maryland, June 1982.

4. THE OBSTACLE DETECTION PROBLEM

A machine cannot understand what it cannot represent. In the domain of autonomous vehicles, a machine cannot avoid obstacles unless it can perceive an obstacle and correctly interpret its perception. This section presents an approach to obstacle perception for autonomous land vehicles.

What is an obstacle? In its most general meaning, an obstacle is a region that a vehicle cannot or should not traverse. Avoiding regions where a vehicle is physically capable of traversing but for some reason should not go (such as not driving the wrong way down a one-way street) would require a level of artificial intelligence that is beyond the scope of this work.

Excluding places where a vehicle can go but shouldn't, one is left with regions that can be defined by their shape and material properties. Rocks, street signs, and steep slopes are all obstacles whose defining characteristic is their shape. Swamps and ice patches on the other hand may have sufficiently flat surfaces for navigation but their material properties make them obstacles for a land vehicle that is not specially equipped. Although material properties are important for determining navigability, they are not readily measured by current remote sensing devices on autonomous vehicles, so for now the simplifying assumption will be made that regions can be adequately categorized by their geometry alone.

4.1. Laser Range Scanners

4.1.1. General Characteristics

There are two types of laser range scanners: time-of-flight (TOF) and phase-shift. TOF scanners measure how much time a laser beam takes to travel from the scanner to an object and back. They operate analogously to sonic scanners except that light travels one million times faster than sound. This creates a significant practical problem of measuring such small units of time. In order to resolve ranges to within 3 inches the detector must be able to accurately measure 170 picosecond units of time.

Phase-shift range scanners modulate the power amplitude of a laser beam and measure the phase difference between the reference wave form and the returning signal. Phase shifts are generally easier to measure than picosecond time differences. The main drawback of phase-shift devices is that one is not measuring an absolute range but rather the true range (up to a multiple of the scanner's half-wavelength). The half-wavelength distance is called an "ambiguity interval" because range measurements are ambiguous by integer increments of the half-wavelength. For example, if the wavelength is 128 feet (i.e. an ambiguity interval of 64 feet) then a range measurement of 1 foot means that the true range to the object is 1 foot, or 65 feet, or 129 feet, etc. In practice, this is a serious problem only for hilly terrain. Note that the wavelength that determines the ambiguity interval is the power modulated wave, not the underlying electromagnetic light wave. The underlying light has an extremely short wavelength, on the order of microns, which gives the scanner its excellent directional resolution.

Older range scanners were seriously hindered by the length of time they needed to form a complete range image. In 1977, [Nitzan] built a phase-shift device that took two hours to acquire a 128x128 image. By 1983 [Jarvis] reported a TOF scanner that could acquire a noisy 64×64 image in four seconds. A current state of the art range scanner is built by the Environmental Research Institute of Michigan (ERIM). It uses a phase-shift design to achieve 0.5 second acquisition rates for 8 bit 64×256 images.

4.1.2. The ALV Range Scanner

The processing of range images is affected by the details of the laser scanner that produces the image. This section describes the ERIM range scanner that is used by the Autonomous Land Vehicle project. More details are available in [Larrowe]. The parameters of the scanner are the result of optimizing many, often conflicting, features including power requirements, ability to penetrate atmospheric moisture, and safety hazards. The evolution of these features is described in [Zuk].

A 100 mW laser generating 0.82 μm wavelength radiation is at the core of the range scanner. Modulation of the laser's power source creates a sinusoidal wave whose frequency (f) is 7.684 MHz. The scanner measures the phase shift (Δp) of the reflected laser beam. If ρ is the range to an object and Δt is the time the signal takes to travel the round-trip from the laser to the object and back then

$$\Delta p = 2\pi f \Delta t \tag{1}$$

and

$$\Delta t = \frac{2\rho}{c} \quad \text{where } c \text{ is the speed of light} \tag{2}$$

Combining equations (1) and (2) yields

$$\rho = \frac{\Delta p \ c}{4\pi f} \tag{3}$$

However, the measured Δp is the true Δp modulo 2π so the largest measurable value of Δp is less than 2π. Applying the frequency and the upper limit of Δp to equation (3) results in the largest measurable ρ having an upper limit of 64 feet. Therefore, 64 feet is the ambiguity interval of the ERIM scanner.

The intensity of the return signal is proportional to both the inverse square of the range and the reflectivity of the surface that reflects the signal. Most natural objects have sufficiently similar reflectivities that the intensity could be used to determine how many ambiguity intervals must be added to obtain the absolute range. Unfortunately, bare metal parts and specially reflective materials on street signs and road stripes render this method unusable for most situations.

Ambient sources of 0.82 μm light do not interfere with the return signal's measurement because the incoming light is filtered to remove radiation that is not modulated at the laser's broadcast frequency of 7.684 MHz. It is extremely unlikely that natural sources would be modulated at this frequency (intentional interference in an adversarial situation is of course possible).

Figure 4.1 (from [Larrowe]) shows the major components of the scanner. The nodding mirror determines the plane that the beam will be in for any given

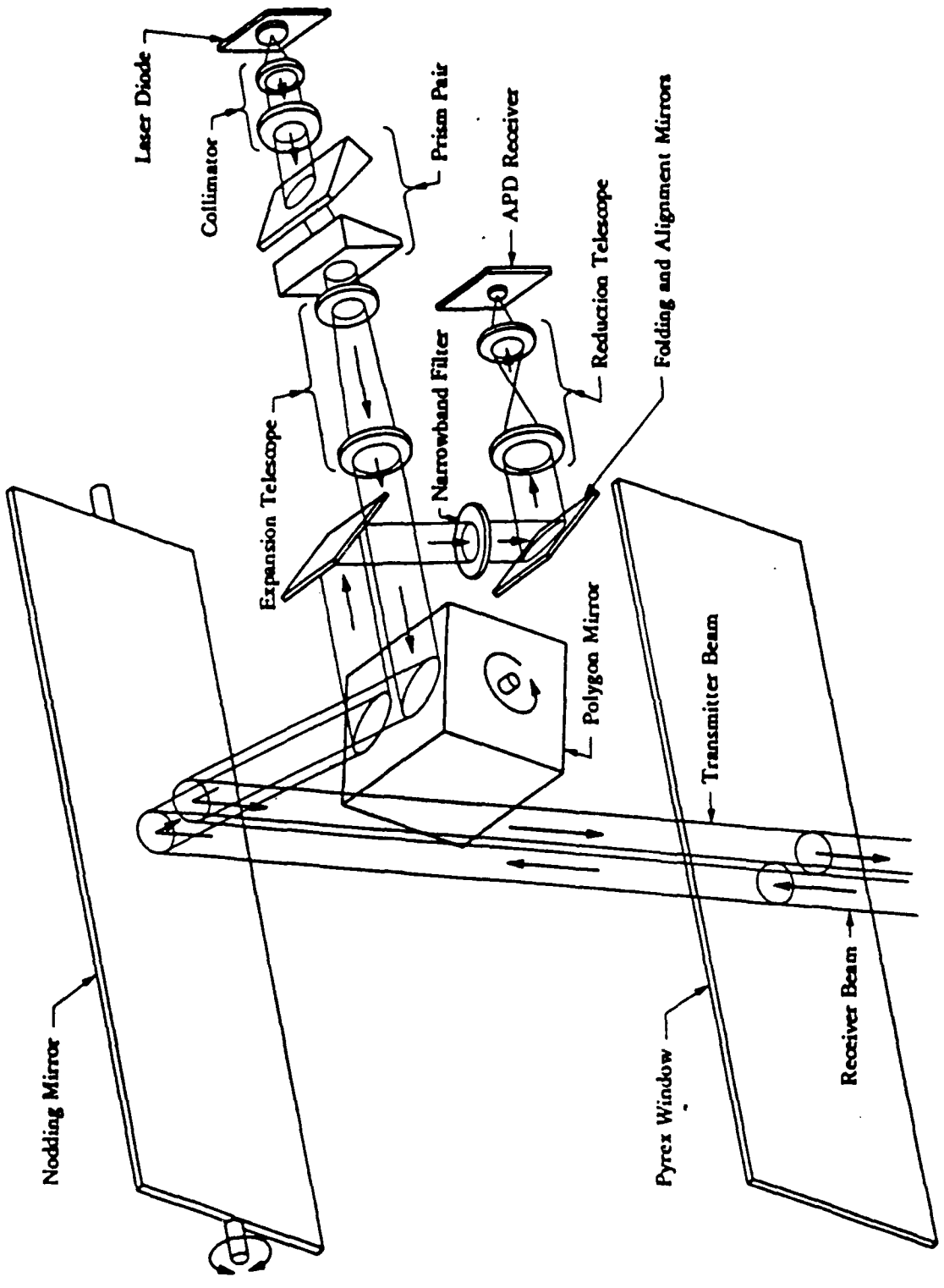


Figure 4.1: The ERIM Range Scanner (from [Larrowe]).

row of the range image. The angle within the plane is controlled by the rotating polygon mirror. Figure 4.2 illustrates the spherical coordinate system (θ, ϕ, ρ) that naturally describes a range image. The scanner, which is mounted on the ALV approximately nine feet above the ground, is at the origin (O) of the system. The positive Y axis points directly down toward the ground. The positive Z axis points out in the direction that the ALV is currently travelling. The length of the line segment OM is the range (ρ) to the point M.

The right triangle Ocb is in the YZ plane. $\angle cOb$ forms the vertical scan angle, ϕ. Each row in a range image is taken from a plane that contains the X axis and is ϕ degrees beneath the Z axis. The rectangle $OaMb$ is in this plane. $\angle aOM$ forms the horizontal scan angle, θ. Each column in a range image corresponds to a particular θ. This geometry results in the following relationships:

$$x = \rho\cos(\theta) \tag{4}$$

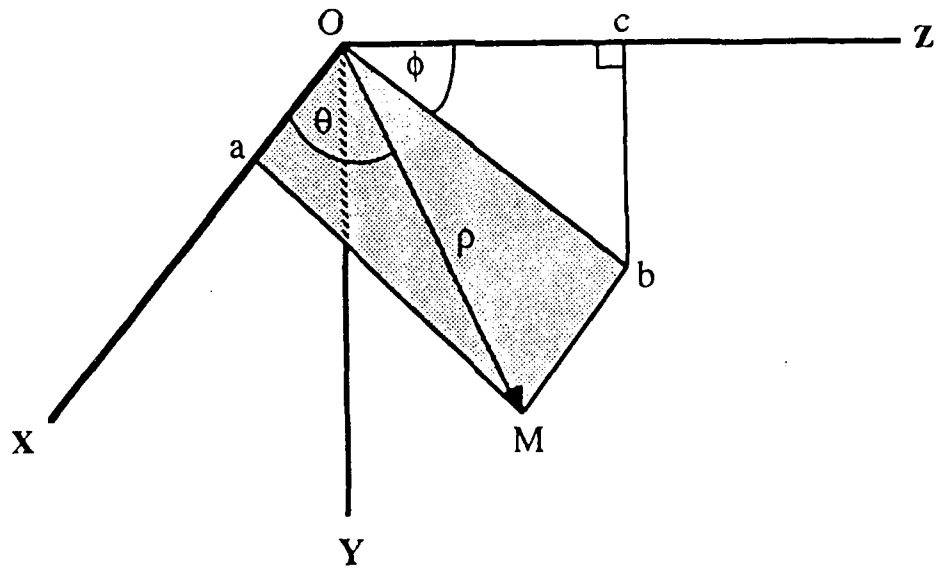
$$y = \rho\sin(\theta)\sin(\phi) \tag{5}$$

$$z = \rho\sin(\theta)\cos(\phi) \tag{6}$$

The 64 rows of the image are at equally spaced values of ϕ, and the 256 columns are at evenly spaced values of θ. An ERIM range image has a 30 degree vertical field of view in which ϕ goes from approximately 6 degrees to 36 degrees. The 80 degree horizontal field of view extends from a θ of 130 degrees to a θ of 50 degrees. Although the total magnitudes of the fields of view are fixed, the orientations can be altered either by internal controls in the scanner or by moving the external platform that the scanner is mounted on.

The ERIM scanner has a vertical sampling interval of 0.3125 degrees and a horizontal sampling of 0.46875 degrees. Since the laser beam has an angular divergence of 0.5 degrees, a scene is densely sampled. This removes the need for sophisticated interpolation techniques such as those proposed by [Boult] or [Choi] for sparse range data.

Multiple objects at various ranges may occur within the 0.5 degree solid cone that forms a single pixel's field of view. The signal that returns to the scanner will indicate a range that is a complex average of all the ranges encountered within the cone. For example, if half of a cone intercepts a tree and the other half travels on to the ground then the returning signal would yield a value that is somewhere between the distance to the tree and the most distant ground that is



ρ = range
ϕ = vertical scan angle
θ = horizontal scan angle

Figure 4.2: Range Image Coordinate System

within the cone. This is called the mixed pixel problem. A strategy for avoiding range errors resulting from mixed pixels is presented in Section 4.4.

Due to the ambiguity effect, output range values are all between zero and 64 feet. They are quantized into three-inch units so that the final output of the ERIM scanner is a 64×256 array of 8 bit values ranging from zero to 255.

4.2. Early Processing for Range Images

There have been many proposed approaches for the initial segmentation of range images. Early work by [Duda] involved converting a pixel's spherical coordinates into Cartesian coordinates and then extracting horizontal planes by histogramming the vertical coordinate of each pixel. Vertical planes were found by projecting points onto the ground plane and detecting lines with Hough transforms. Pixels that were not found to be in vertical or horizontal planes were grouped by similar gray levels. [Parvin] also converted each pixel's spherical coordinates to Cartesian coordinates but then $\partial y / \partial x$ and $\partial y / \partial z$ were estimated at each pixel. Pixels were then grouped into common planes if their derivatives were sufficiently similar. Parvin reported that applying this procedure to a 128×128 range image took 4.5 minutes on a VAX 11-750, much too long for navigation purposes.

The most common approach for initial segmentation is probably the fitting of surface normals to each pixel. [Hebert] calculated surface normals and represented them on a Gaussian sphere so that planes appeared as points (or clusters), cylinders were circles, and other 3-dimensional shapes could be similarly

extracted. [Sethi] projected surface normals onto an image plane and found equi-magnitude contours. [Yang] did the same but extended the analysis by also finding equi-orientation contours. [Jain, Hoffman, Hoffman] all used surface normals to classify surfaces as being planar, convex, or concave as the first step in object recognition algorithms. [Lin] extracted surface patches with common surface normals to match against possible models in another object recognition scheme.

Higher order polynomials have also been fitted to range data. [Hall] characterized surface patches in synthetic range images by their quadric coefficients. [Jain], however, reported that quadric fits did not work well for real range images. [Sharma] compared quadric and planar fits on outdoor range imagery for extracting road edges and found no advantages to using the slower quadric method. [Olivier] proposed a least-squares cubic fit and [Vemuri] used splines under tension.

[Inokuchi] applied a ring operator to range images in conjunction with a discrete Fourier transform to generate amplitude and phase information for characterizing pixels.

Outside of projects related to the Autonomous Land Vehicle, little work has been reported on outdoor range images. The investigations previously mentioned in this section, with the exception of [Jain, Sharma, Hebert], were all done using synthetic range images or indoor scenes with very fine gradations. The range images used by [Hoffman], for example, had an ambiguity interval of approximately eight inches and a range resolution of 0.03125 inches. This level of fine

resolution allows the use of more sensitive algorithms than are applicable to outdoor range data where the ambiguity interval is 64 feet and the resolution is theoretically three inches but, in practice, usually closer to six inches. The noise and the noncontinuous nature of outdoor scenes may make methods more complex than planar surface fitting unsuitable for outdoor navigation. In addition, complex techniques take longer to run which can severely limit the speed at which they can control a moving vehicle.

[Sharma] and [Hebert] have shown that fitting planes can be used for low-level processing of outdoor range images, but even simpler algorithms may also be adequate. The fastest of the ALV obstacle detection algorithms, range differencing, simply subtracts the range image of an actual scene from the expected range image of a flat plane. While rapid, this technique is not very robust. Small errors in the orientation of the scanner or a mild slope in the land will result in false indications of obstacles. We propose using the first derivatives of the range with respect to the vertical and horizontal scan angles as an improved, fast obstacle detector. The details of this algorithm, an analysis of its robustness, and the results using actual outdoor range data are given in the following sections.

4.3. The Range Derivative Algorithm for Obstacle Detection

4.3.1. Central Ideas

When deciding if a surface is an obstacle or not the pertinent feature is the change in height across the surface. If the change is too rapid then the surface is

unnavigable. A surface normal contains the necessary information on the change in height but calculating surface normals is computationally intensive. The surface normal at a point is a function of $\partial y/\partial x$ and $\partial y/\partial z$. Simply calculating the slope, $\partial y/\partial z$, would provide significant information concerning a surface's navigability. However, computing the slope directly from a range image is not much easier than calculating a surface normal. What can be done very quickly, though, is finding $\partial\rho/\partial\theta$ and $\partial\rho/\partial\phi$. See Figure 4.2. The following section first shows how $\partial\rho/\partial\phi$ can be closely linked to $\partial y/\partial z$ and then how $\partial\rho/\partial\theta$ can be a measure of $\partial y/\partial x$. Using our knowledge of how range derivatives reflect changes in height across a surface we can then design a rapid obstacle detection algorithm.

The differential of a function $y(\phi,\rho,\theta)$ can be written as

$$dy = \frac{\partial y}{\partial \phi}\, d\phi + \frac{\partial y}{\partial \rho}\, d\rho + \frac{\partial y}{\partial \theta}\, d\theta \tag{7}$$

If θ is held constant so that the $d\theta$ term is zero then equation (7) applied to equation (5) gives

$$\Delta y = \rho \sin\theta \cos\phi\, \Delta\phi + \sin\theta \sin\phi\, \Delta\rho \tag{8}$$

where the infinitesimal terms dy, $d\phi$, and $d\rho$ have been replaced by their finite Δ equivalents. In a similar fashion, equation (6) can be differentiated to yield

$$\Delta z = -\rho \sin\theta \sin\phi\, \Delta\phi + \sin\theta \cos\phi\, \Delta\rho \tag{9}$$

Dividing Δy by Δz yields

$$\frac{\Delta y}{\Delta z} = \frac{\rho\cos\phi\,\Delta\phi + \sin\phi\,\Delta\rho}{-\rho\sin\phi\,\Delta\phi + \cos\phi\,\Delta\rho}$$

$$= \frac{\frac{\Delta\rho}{\rho}\frac{\tan\phi}{\Delta\phi} + 1}{\frac{\Delta\rho}{\rho}\frac{1}{\Delta\phi} - \tan\phi} \tag{10}$$

If ϕ is held constant then equation (7) becomes

$$\Delta y = \sin\theta\sin\phi\,\Delta\rho - \rho\sin\phi\cos\theta\,\Delta\theta \tag{11}$$

and equation (4) can be differentiated to obtain

$$\Delta x = \cos\theta\,\Delta\rho - \rho\sin\theta\,\Delta\theta \tag{12}$$

Dividing equation (11) by equation (12) and regrouping yields

$$\frac{\Delta y}{\Delta x} = \frac{\frac{\Delta\rho}{\rho}\frac{\tan\theta}{\Delta\theta}\sin\phi - \sin\phi}{\frac{\Delta\rho}{\rho}\frac{1}{\Delta\theta} - \tan\theta} \tag{13}$$

Excluding the terms in equations (10) and (13) that we know a priori, we see that the changes in height in the x and z directions are a function of $\Delta\rho/\rho$. If we used some approximation of ρ we would have a direct relationship between the easily calculated $\Delta\rho$ for a fixed θ or ϕ at a pixel and the slopes at that pixel. Our experiments with real range data suggest that the following is an adequate approximation:

$$\rho \approx \frac{H}{\sin\theta\sin\phi} \tag{14}$$

where H is the height of the range scanner above the ground. Equation (14)

comes from substituting H for y in equation (5). In hilly terrain this approximation is probably not adequate but it works well for many scenes and it will be shown in the next section that the derivative algorithm that uses this approximation is less sensitive to orientation errors than other algorithms of similar simplicity and speed.

Using equation (14) we can calculate what $\Delta\rho$ would be at each pixel if the slopes were zero. The difference between this predicted $\Delta\rho$ and the actual $\Delta\rho$ found in a range image is a measure of the actual slope. Large differences between predicted and actual $\Delta\rho$'s will be formed by edges of objects as well as surfaces with steep slopes. Thresholding the absolute values of these differences yields pixels that are likely to be on obstacles.

One could of course simply threshold the actual $\Delta\rho$'s without first subtracting the expected $\Delta\rho$'s and assume that large $\Delta\rho$'s indicate surfaces that have steep slopes and hence are not navigable. This approach, however, would severely reduce one's capability to detect obstacles. A perfectly flat surface will yield a $\Delta\rho$ of about 10 if it is 60 feet away but the same surface at a range of 10 feet only has a $\Delta\rho$ of about 0.3. This wide range in $\Delta\rho$'s leaves any thresholding algorithm in a bind. Small threshold levels would find nearby obstacles but more distant flat surfaces would be falsely labelled as obstacles. Conversely, larger threshold levels would hide significant obstacles that are near the range scanner. What is needed is a variable threshold setting. This approach points out another way of looking at the range derivative algorithm: we are, in essence, creating a variable threshold that changes across an image based on

expected $\Delta\rho$'s. While this simplisti view is a useful description, the derivative algorithm is founded on the mathematical relationships between $\Delta\rho$ and a surface's slopes and is not a randomly chosen heuristic for setting variable threshold levels.

4.3.2. Geometrical View of the Algorithm

Equation (10), which showed the relationship between $\Delta\rho$ and a surface's slope, was derived from the differential equations for dy and dz. Figure 4.3 illustrates the following description of how equation (10) can also be derived from the geometry of a range image in the neighborhood about a pixel. We wish to

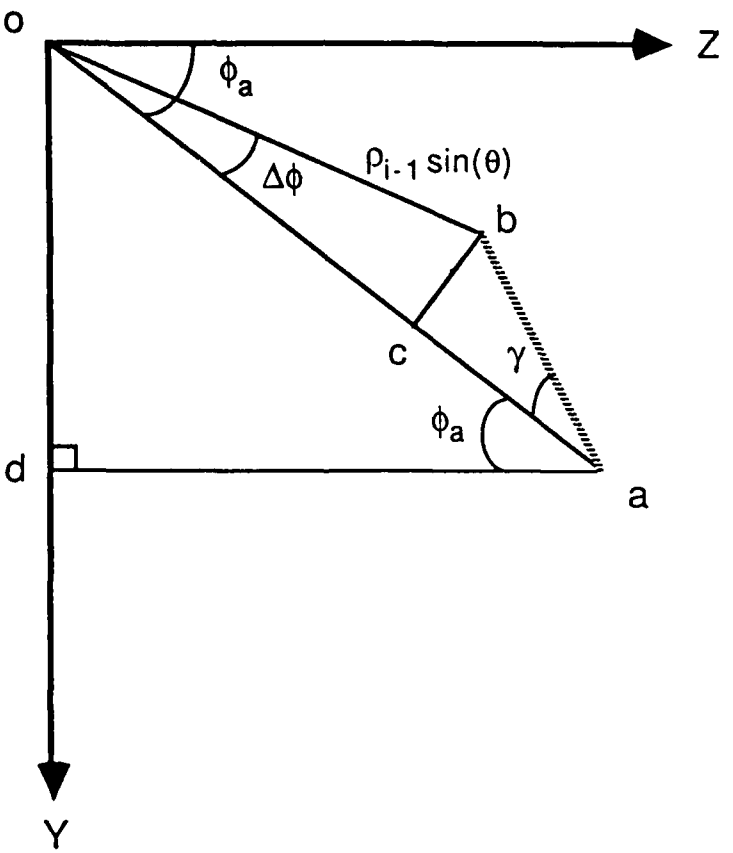


Figure 4.3: The Geometry of the Range Derivative Algorithm

estimate $\Delta y / \Delta z$ at some Range$[i,j]$ in an image. Consider projecting Range$[i-1,j]$ and Range$[i+1,j]$ into the YZ plane. Their projections would be the line segments, respectively, $\overline{ob}$ and $\overline{oa}$ as shown in Figure 4.3. If $\overline{da}$ is a line segment that is parallel to the Z axis then the tangent of $\angle dab$ is equal to $\Delta y / \Delta z$ between Range$[i-1,j]$ and Range$[i+1,j]$. If $\Delta\phi$ is small then the tangent of $\angle dab$ is likely to be a reasonable estimate of the slope at Range$[i,j]$. This tangent can be found in the following manner. As Figure 4.3 shows,

$$\angle dab = \phi_a + \gamma \qquad \text{where } \phi_a = \phi_{row=i+1}$$

and

$$\Delta y / \Delta z = \tan(\angle dab) = \tan(\gamma+\phi_a).$$

Applying the tangent summation formula to the last equation gives us

$$\Delta y / \Delta z = \frac{\tan\gamma + \tan\phi_a}{1 - \tan\gamma\tan\phi_a} \tag{15}$$

If $\angle bca$ were 90 degrees then $\tan\gamma = |\overline{bc}| / |\overline{ca}|$ (where $|\overline{bc}|$ is the length of segment $\overline{bc}$). $\angle bca$ is actually equal to $90 + \Delta\phi/2$ but $\Delta\phi/2 = 0.476$ degrees so we can reasonably approximate $\angle bca$ by 90 degrees which gives us

$$\tan\gamma \approx \frac{|\overline{bc}|}{|\overline{ca}|} \tag{16}$$

By definition, c is the point on $\overline{oa}$ where $|\overline{oc}| = |\overline{ob}|$. This means that $|\overline{ca}| = |\overline{oa}| - |\overline{ob}|$. Since $\overline{oa}$ and $\overline{ob}$ are projections of ranges into the YZ plane we know that

$$| \overline{oa} | = \text{Range}[i+1,j] \sin\theta = \rho_{i+1} \sin\theta \tag{17.1}$$

and

$$| \overline{ob} | = \text{Range}[i-1,j] \sin\theta = \rho_{i-1} \sin\theta \tag{17.2}$$

So,

$$| \overline{ca} | = \Delta\rho \sin\theta \qquad \text{where} \Delta\rho = \rho_{i+1} - \rho_{i-1} \tag{17.3}$$

If we draw a line from the origin to the midpoint of $\overline{bc}$ we will bisect $\Delta\phi$ and have a right triangle in which $\overline{ob}$ is the hypotenuse and $\overline{bc} / 2$ is the length of the side that is opposite of the $\Delta\phi / 2$ angle. Hence we know that

$$\frac{| \overline{bc} | /2}{| \overline{ob} |} = \sin(\Delta\phi/2) \tag{18}$$

But $\Delta\phi/2$ is quite small so $\sin(\Delta\phi/2) \approx \Delta\phi/2$. Applying this approximation and equation (17.2) to equation (18) yields

$$| \overline{bc} | = \rho_{i-1} \sin\theta \, \Delta\phi \tag{19}$$

Equations (17) and (19) show us that equation (16) can be restated as

$$\tan\gamma = \frac{\rho_{i-1} \sin\theta \, \Delta\phi}{\Delta\rho \sin\theta} = \frac{\rho_{i-1} \Delta\phi}{\Delta\rho} \tag{20}$$

Combining equations (15) and (20) and rearranging terms yields

$$\frac{\Delta y}{\Delta z} = \frac{\dfrac{\Delta\rho}{\rho_{i-1}} \dfrac{\tan\phi_a}{\Delta\phi} + 1}{\dfrac{\Delta\rho}{\rho_{i-1}} \dfrac{1}{\Delta\phi} - \tan\phi_a} \tag{21}$$

Comparing equation (21) to equation (10) we see that the two approaches for relating $\Delta y / \Delta z$ to $\Delta\rho$ yield the same result. A similar correlation could also

be made for $\Delta y / \Delta x$ and the projections into the XY plane of Range[i,j+1] and Range[i,j−1].

4.3.3. Sensitivity to Scanner Perturbations

To know the shape of the world from a range image, it is first necessary to know where the range scanner is and what its orientation is. When an ERIM range scanner is mounted on the ALV, the scanner's location (especially its height above the ground) can be found with sufficient accuracy. Its orientation, on the other hand, has been surprisingly difficult to measure. Martin Marietta, the integrating contractor for the ALV project, has reported that determining the roll, pitch, and yaw of the scanner has been a serious continuing problem. For this reason it is useful to study the sensitivity of the range derivative algorithm to errors in scanner orientation and to compare this sensitivity to that of the range difference and height difference algorithms. For height differencing, one first converts the range image to Cartesian coordinates and then differences the expected height (y coordinate) instead of the range.

There are many ways one could measure errors in obstacle detection algorithms. We chose to consider what happens when various algorithms are applied to the range image of a flat plane in which the image was taken by a scanner rotated in some manner. The three algorithms studied were: our range derivative algorithm (broken down into the θ derivative and the ϕ derivative steps), the height difference algorithm, and the range difference algorithm. The output of these algorithms when applied to an image of a flat plane (i.e. a plane parallel to

the XZ plane at a known $Y = H$) should be zero at each pixel. If the range image is taken by a scanner rotated about a particular axis then the results of applying an algorithm to the perturbed image is a good measure of the algorithm's sensitivity to that type of rotation. The following rotations were used to generate perturbed range images:

1) Rotating the scanner about the X axis (i.e. pitch error) by δ_x so that

$$\rho = \frac{H}{\sin\theta \sin(\phi+\delta_x)} \tag{22}$$

2) Rotating the scanner about the Y axis would not alter the range image because images of flat planes are invariant to this type of rotation (this of course is not generally true for planes that do not have a constant Y value). However, any error in the timing or orientation of the polygon mirror that determines the θ of the laser beam could lead to θ being off by some δ_y (which we will call the yaw error). If the data stream was not properly synchronized one could also believe that the range at some $\theta + \delta_y$ was the range for θ. Yaw error has the form

$$\rho = \frac{H}{\sin(\theta+\delta_y) \sin\phi} \tag{23}$$

3) If the scanner is rotated about the Z axis (roll error) by some δ_z then

$$\rho = \frac{H}{\cos\delta_z \sin\theta \sin\phi + \sin\delta_z \cos\theta} \tag{24}$$

4) The most severe perturbation considered was the combination of roll, followed by pitch, followed lastly by yaw, so that

$$\rho = \frac{H}{\cos\delta_z \sin(\theta+\delta_y)\sin(\phi+\delta_x) + \sin\delta_z \cos(\theta+\delta_y)} \tag{25}$$

Equations (22–25) were used to produce four different range images in which H was assumed to be nine feet. The derivative, height, and range algorithms were applied to each image. When applying algorithms to real outdoor range images it is necessary to average ranges over a neighborhood to suppress noise. For these perturbation sensitivity experiments we used the following unnormalized summations at each pixel$[i,j]$:

For the θ derivative,

$$\Delta\rho = \sum_{k=i-1}^{i+1} \text{Range}[k,j+1] - \sum_{k=i-1}^{i+1} \text{Range}[k,j-1] \tag{26}$$

For the ϕ derivative,

$$\Delta\rho = \sum_{k=j-1}^{j+1} \text{Range}[i+1,k] - \sum_{k=j-1}^{j+1} \text{Range}[i-1,k] \tag{27}$$

For the height derivative,

$$y = \sum_{k=j-1}^{j+1} \text{Range}[i,k]\sin(\theta_{col=k})\sin(\phi_{row=i})$$

For the range derivative,

$$\rho = \sum_{k=j-1}^{j+1} \text{Range}[i,k]$$

Table 4.1 summarizes the results of these experiments. The table contains two entries for each combination of algorithm and perturbation. The top entry is the largest absolute value in the entire image and represents a worst-case scenario. In many scenes, however, the road will be near the center of the image's horizontal field of view and large errors on the periphery are not critical.

Table 4.1: Comparison of Obstacle Detection Algorithms For Sensitivity to Scanner Perturbations

Perturbation	Magnitude of Errors (1)			
	Theta Derivative Algorithm	Phi Derivative Algorithm	Height Difference Algorithm	Range Difference Algorithm
3 degrees in horizontal angle	0.8 (2) 0.3 (3)	1.5 0.4	5.1 1.6	24.7 6.4
3 degrees in roll angle	3.7 1.1	13.6 2.8	21.2 6.0	103.3 23.1
3 degrees in vertical angle	1.1 0.3	17.3 13.8	25.4 25.4	123.7 98.5
3 degrees in each angle	9.7 2.6	53.5 21.3	72.2 38.9	351.4 150.7

(1) The absolute values of the expected $\Delta\rho$ (or ρ or y) from an unperturbed scanner minus the $\Delta\rho$ (or ρ or y) from a scanner that has been rotated in the manner listed in the 'Perturbation' column.

(2) Maximum error in image.

(3) Maximum error in central 30 degrees of image.

This scenario is represented by the bottom entry which is the largest absolute value within the central 30 degrees of the image (i.e. $105 \leq \theta \leq 85$).

Several important trends emerge from Table 4.1. The θ derivatives were very insensitive to all four rotational perturbations. The ϕ derivatives were somewhat more sensitive. When the entire image was considered, the maximum ϕ derivative errors for each rotation were always at least 25% less than the maximum height difference errors. Within the central 30 degrees of the horizontal field of view, the maximum ϕ derivative errors were 45%–75% less than the maximum height difference errors. The range difference algorithm was very sensitive to all forms of rotations. In several instances the range difference errors were a full order of magnitude larger than the derivative errors. These results clearly show that the derivative algorithms are more robust under rotational uncertainties than either the height difference or the range difference algorithms.

For these experiments the true range values without ambiguity intervals were used. Ambiguity intervals were not simulated so as to avoid mixing the separate issues of ambiguity interval compensation and sensitivity to scanner perturbations.

4.4. Implementation of the Range Derivative Algorithm

There are several practical considerations that affect the implementation of a range derivative algorithm. Chief among these are: choosing an accurate $\Delta\rho$, reducing noise in an image, avoiding ambiguity interval errors, and inhibiting errors from mixed pixels. Our approach for these considerations are explained in

this section and the results from applying them to actual range images are presented.

Equations (26) and (27) give the actual equations used to calculate $\Delta\rho$ for the θ and ϕ derivatives, respectively. (The term "ϕ derivative" is used loosely to mean $\Delta\rho$ calculated for a known $\Delta\phi$ while θ is held constant. Similarly, "θ derivative" means $\Delta\rho$ calculated for a known $\Delta\theta$ while ϕ is constant.) For the ϕ derivatives the choice of calculating $\Delta\rho$ across two rows was a compromise between two conflicting goals. On the one hand, one wants $\Delta\phi$ to be as small as possible so as to insure that the resultant $\Delta\rho$ accurately reflects the slope at the pixel $[i,j]$. The smaller $\Delta\phi$ is, the less likely it is that a surface will be sufficiently curved to cause an inaccurate $\Delta\rho$. On the other hand, decreasing the size of $\Delta\phi$ also decreases the magnitude of the $\Delta\rho$'s, which can impair the accuracy of the $\Delta\rho$ measurements. This is especially true for small $\Delta\rho$'s due to the quantization of range measurements into three inch units. The choice of using $\Delta\theta$ across two columns for the θ derivatives was made for the same reasons.

The summation of the ranges in a three-column neighborhood for the ϕ derivative (and a three-row neighborhood for the θ derivative) has been found to be an effective averaging method. A more sophisticated smoothing operator, the Symmetric-Nearest-Neighbor operator ([Harwood]), was also tested on range images but it did not significantly improve obstacle detection.

The ambiguity interval problem has been approached in two different ways. For actual range images taken from the ALV we have found that if the vehicle is not travelling over very hilly territory, simply deleting the upper few rows of the

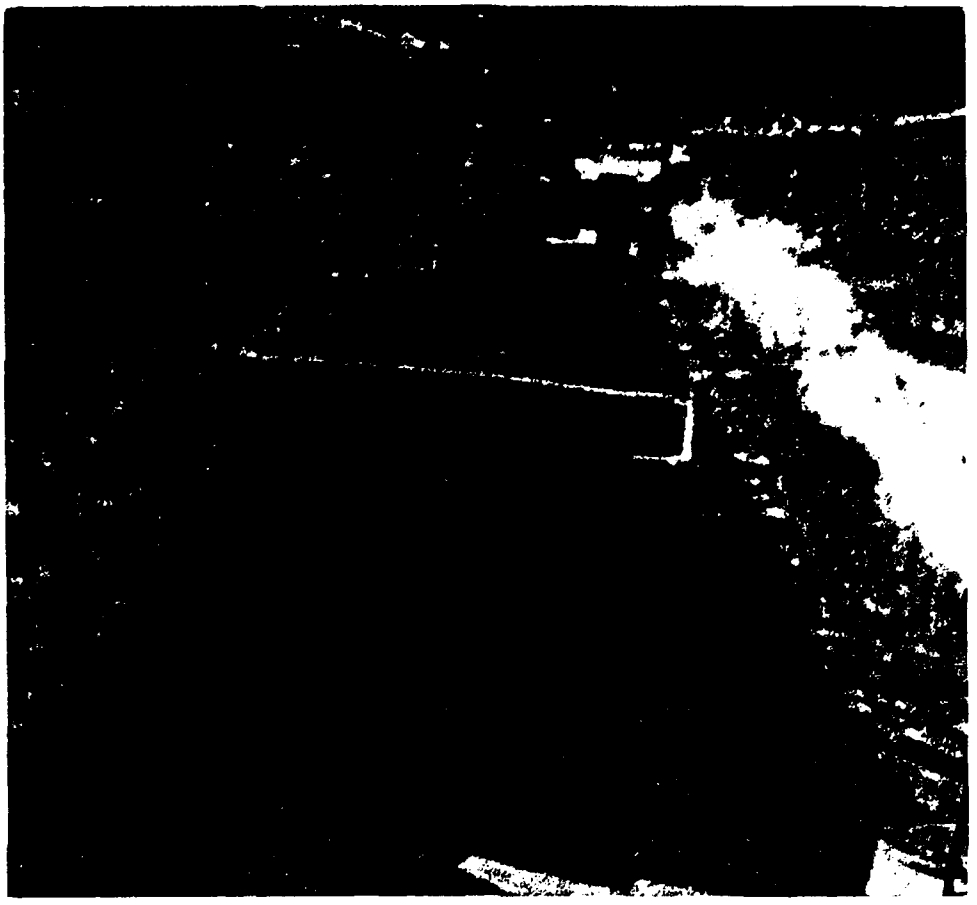


Figure 4.4: Video Image of Road Segment with Carton

image removes most of the pixels that are beyond the first ambiguity interval ($\geq$ 64 feet). This does not affect the obstacle detection algorithm significantly because the laser beam has spread out into a relatively wide cone by the time it has travelled beyond 50–60 feet, resulting in mixed pixels that are of little use for accurate obstacle detection. For example, at a range of 55 feet, a laser beam whose central axis strikes a planar surface at an incident angle of 5 degrees will form an elliptical footprint with major and minor axes, respectively, of 5.5 feet and 0.5 feet.

A more time consuming but somewhat more precise approach has been to examine each column from bottom to top in the image. Whenever adjacent pixels go from large values suddenly to very small values it is reasonable to assume that an ambiguity interval has been reached and that all pixels in the column beyond this point should have an additional 256 added to their ranges. This approach was used in the ALV Simulator that is described in the next section.

Figure 4.4 is a visual picture of Martin Marietta's ALV test track in Denver, Colorado. The picture was taken by the video camera that is mounted on the ALV for visual navigation. Figure 4.5 is a comparison of the fields of view of the video camera and the range scanner. It is an outline of how much territory each device can 'see' on a flat road in a single frame. The camera's view extends further but is narrower than the range scanner's view. Since the camera's vertical field of view goes above the horizon, its view theoretically extends to infinity. The range scanner's view begins several feet closer to the vehicle than does the camera's. This can be clearly seen in Figure 4.6a: a montage of four range images in which the top image was taken by the ALV at the same time as Figure 4.4. Moving down from the top of the montage, each image was taken five feet further down the road. The carton in the lower right corner of Figure 4.4 can be seen in the top two images of Figure 4.6a. The cone that is on the right side of the road about half way up Figure 4.4 is present in all four range images. The cone is not very apparent in Figure 4.6a but it is clearly marked as an obstacle in Figure 4.6b, which is the thresholded output of the θ derivative algorithm. Figure 4.6c is the thresholded output of the ϕ derivative algorithm. The small, dark

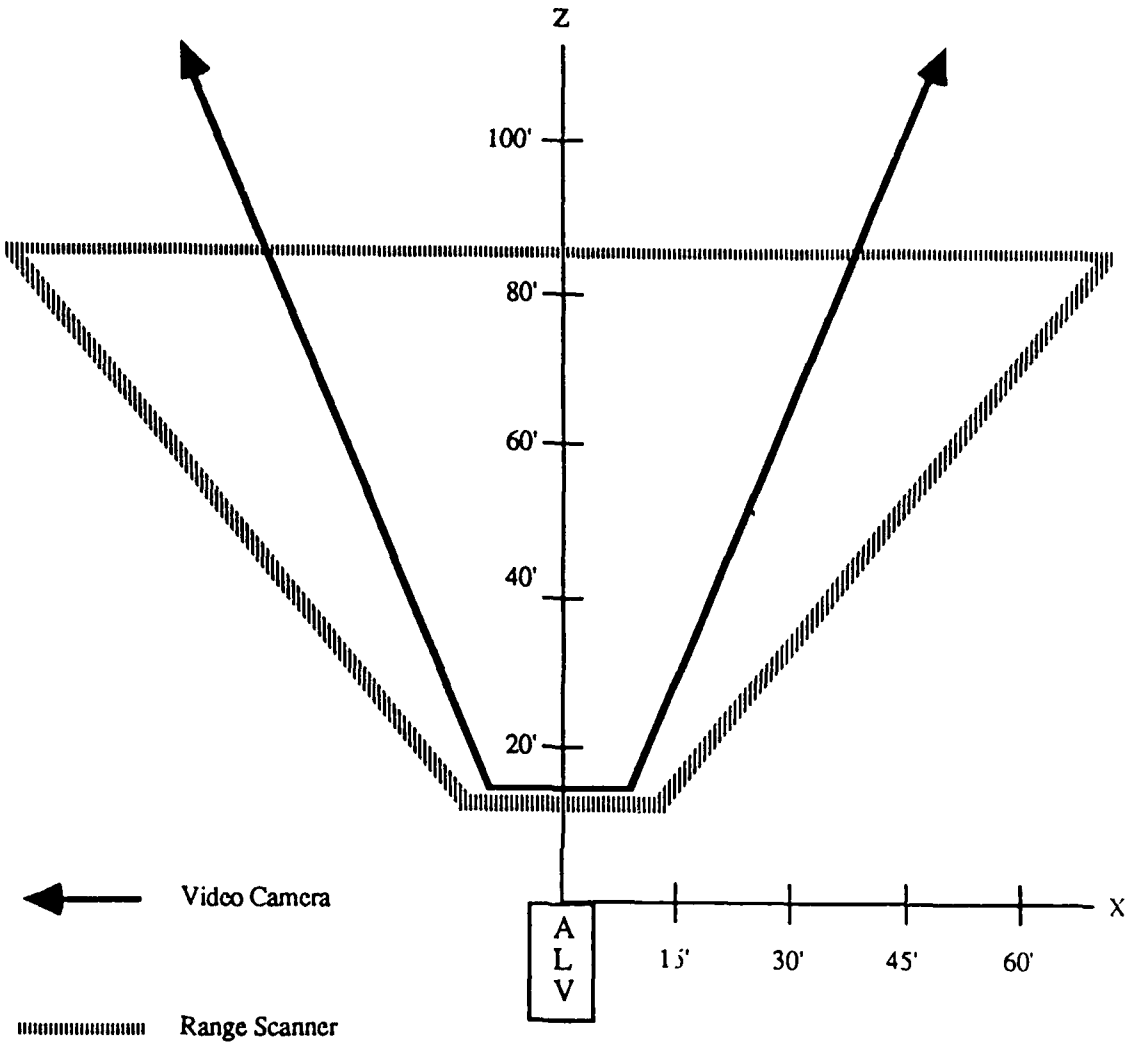


Figure 4.5: Field of View from the ALV of a Flat Road—Range Scanner vs. Video Camera

blob that is located in the center columns of the top few rows of each image in Figure 4.6a and that is marked as an obstacle in Figures 4.6b and 4.6c does not correspond to any actual object in the scenes but rather is spurious data generated by the range scanner's electronics.

All of the bina.y images in this section were thresholded manually. Automatic thresholding was not a topic of research so while the effect of differing

threshold levels was observed, no attempt was made to determine how optimal levels should be set.

Figure 4.6d is the result of two processing steps. First, the binary images 4.6b and 4.6c were ORed together and then isolated obstacle pixels (i.e. pixels that had no 8-way neighbors marked as obstacles) were deleted.

Location errors due to mixed pixels can be minimized by using pixels that come from the interior of an obstacle and avoiding pixels at the edges. If one

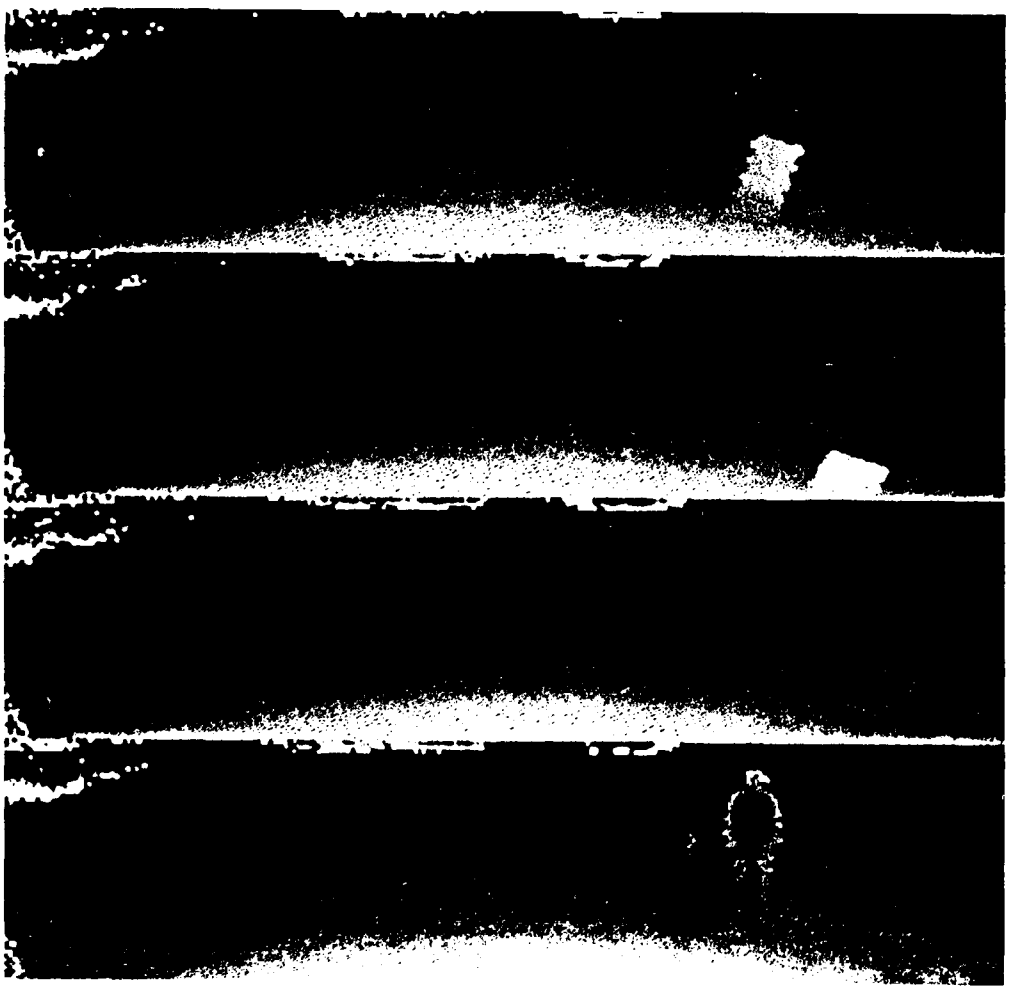
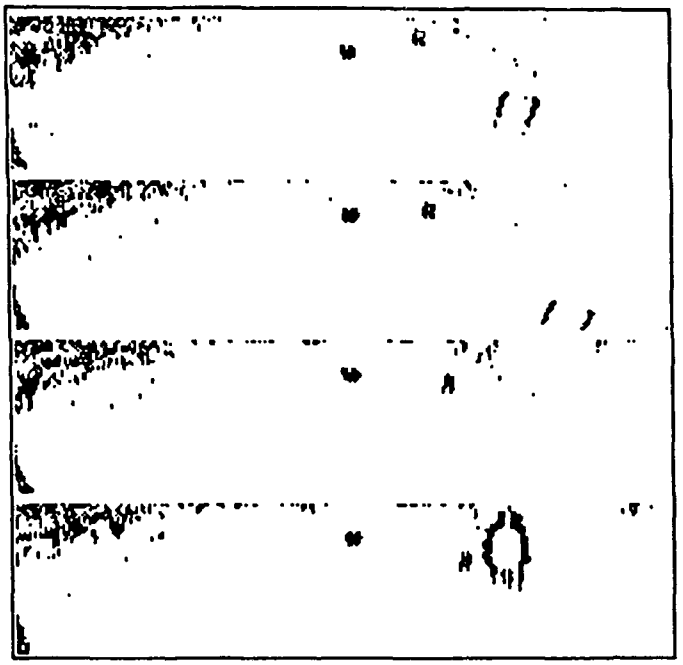


Figure 4.6a: First Montage of Range Images

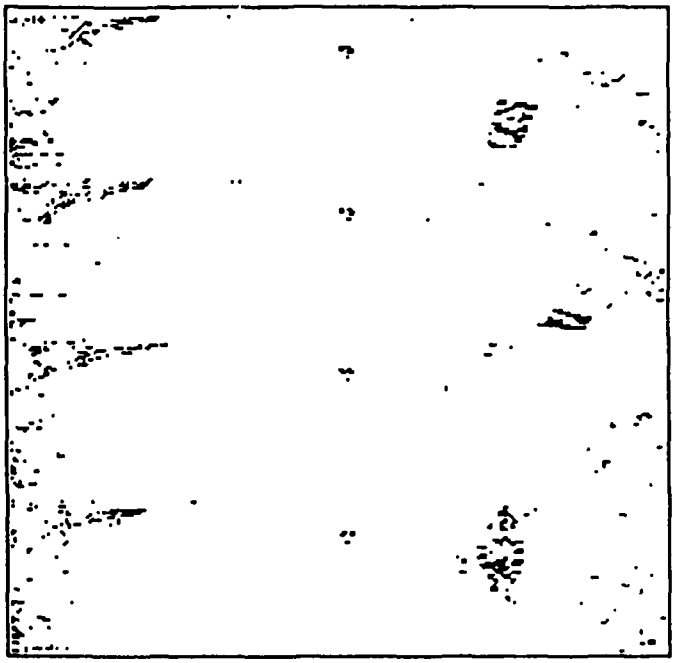
assumes that obstacle surfaces are usually not concave then the sign of the derivative at an edge pixel indicates the direction of the center of the obstacle. For ϕ derivatives (as defined in equation (27)), negative $\Delta\rho$'s occur at the top edges of obstacles. Positive $\Delta\rho$'s occur at bottom edges. For θ derivatives (as defined in equation (26)), left edges have negative $\Delta\rho$'s and right edges have positive values.

Our implementation of this strategy has two parts. First, if an obstacle pixel is at location $[i,j]$ in the image, use the sign of the derivative to determine which direction is likely to be away from the edge and toward the interior. The direction determines which of the adjacent pixels will be used for associating a location with the obstacle found at $[i,j]$. If, for example, $[i,j+1]$ is determined to be toward the obstacle's center then the three-dimensional location of the pixel $[i,j+1]$ in the range image will be used to place the obstacle that was found at $[i,j]$ in the thresholded image.

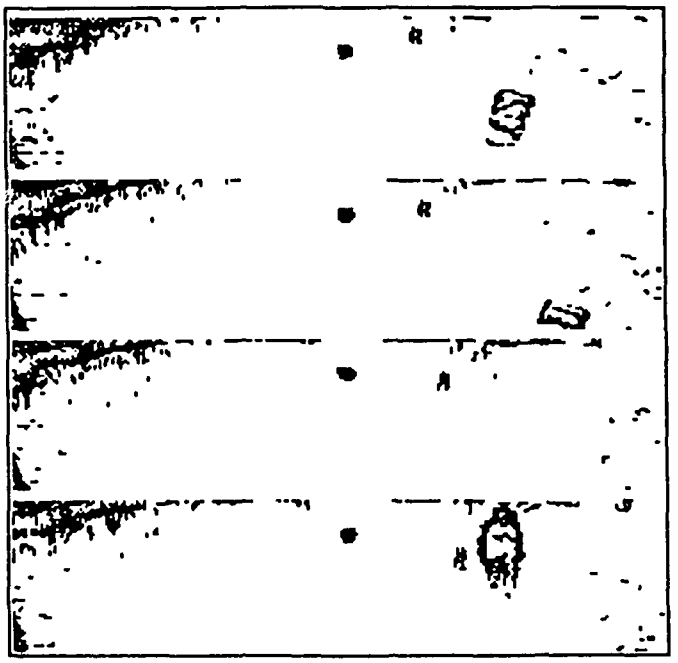
Figures 4.7–4.9 show the improved positioning of obstacle pixels achieved by this mixed-pixel minimization algorithm. Figure 4.7 is the video image of a box located 25 feet in front of the ALV. A montage of range images in which the box is 25, 30, 35, and 40 feet from the ALV is given in Figure 4.8a. The obstacle pixels from the combined θ and ϕ derivatives are shown in Figures 4.8b and 4.8c. In Figure 4.8b, the pixels have been shifted toward the interior by the algorithm while in Figure 4.8c they have not been shifted. The projection onto the ground plane of the obstacle pixels is shown in Figure 4.9a (with shifting) and in Figure 4.9b (without shifting). The box's pixels are circled in both of these figures. The



(b)



(c)



(d)

Figure 4.6b–d: (b) Thresholded θ Derivatives of First Montage. (c) Thresholded ϕ Derivatives of First Montage. (d) Combined θ and ϕ Derivatives of First Montage.

unshifted pixels are clearly more scattered than the shifted pixels. The trapezoid in Figures 4.9a and 4.9b is an approximate outline of the scanner's field of view, similar to the trapezoid in Figure 4.5. Figures 4.10a and 4.10b are the ground projections of the box at 40 feet with and without shifting, respectively.

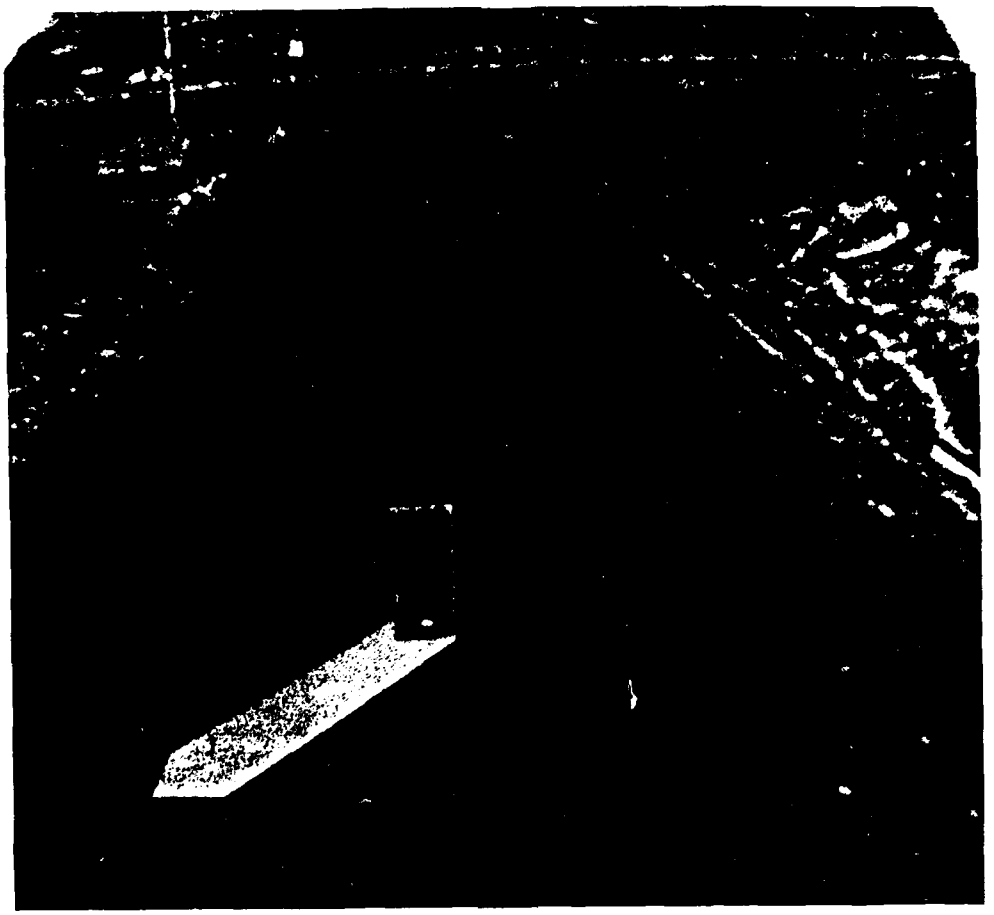


Figure 4.7: Video Image of Road Segment with Box

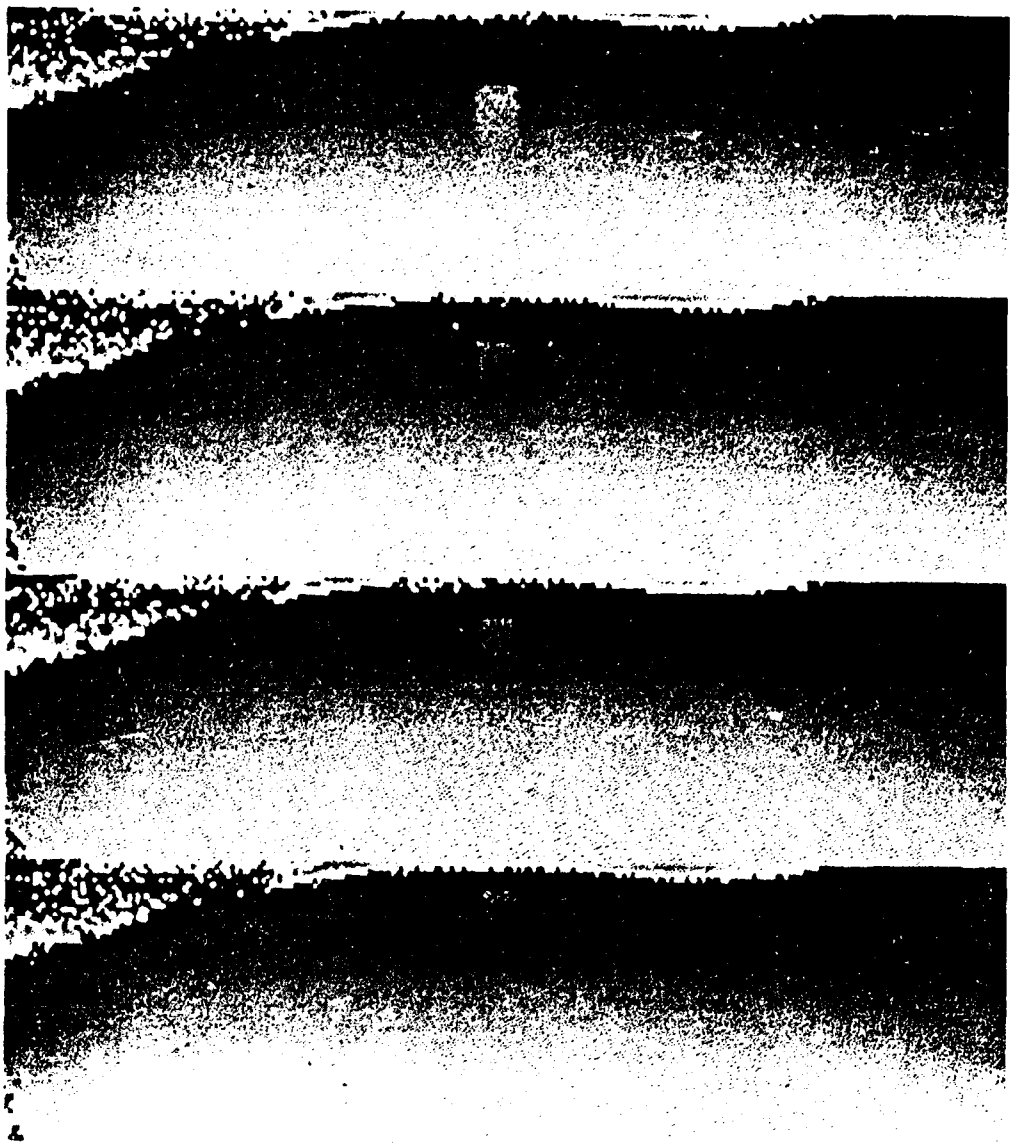


Figure 4.8a: Montage of Box Segment Range Images

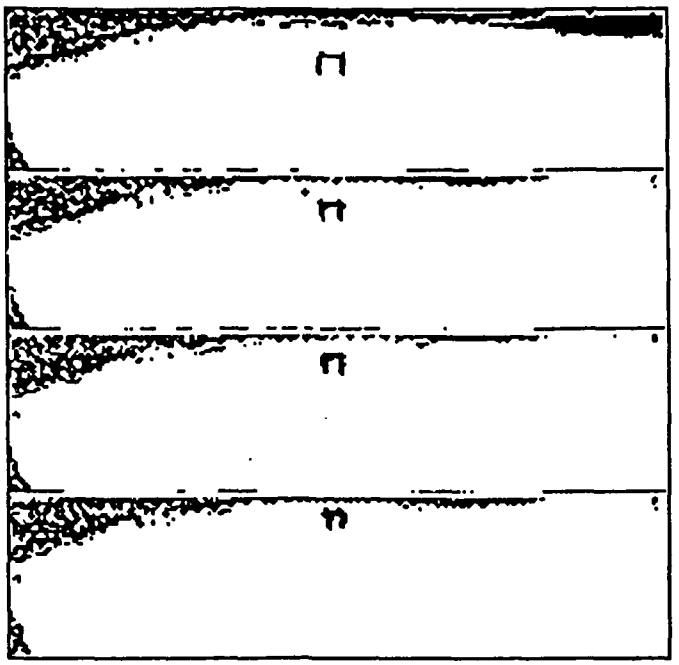


Figure 4.8b: Obstacle Pixels of Box Segment Montage with Mixed Pixel Minimization

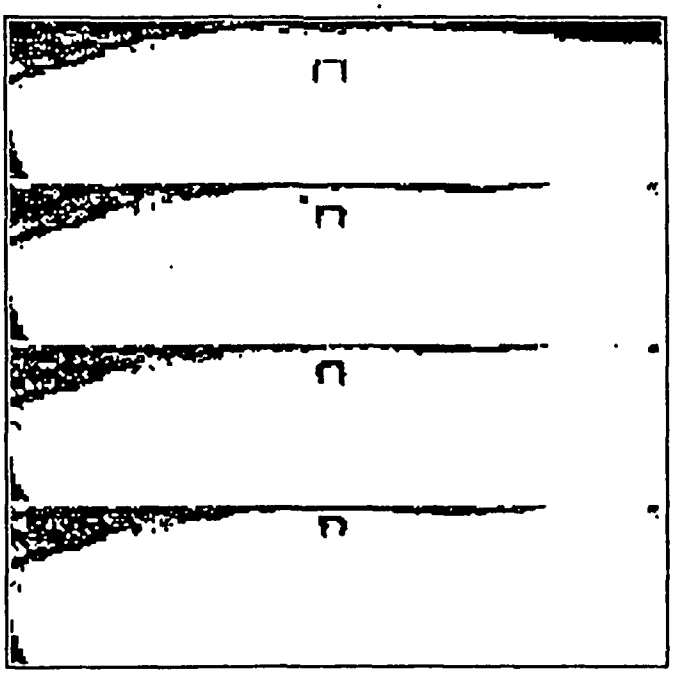


Figure 4.8c: Obstacle Pixels of Box Segment Montage without Mixed Pixel Minimization

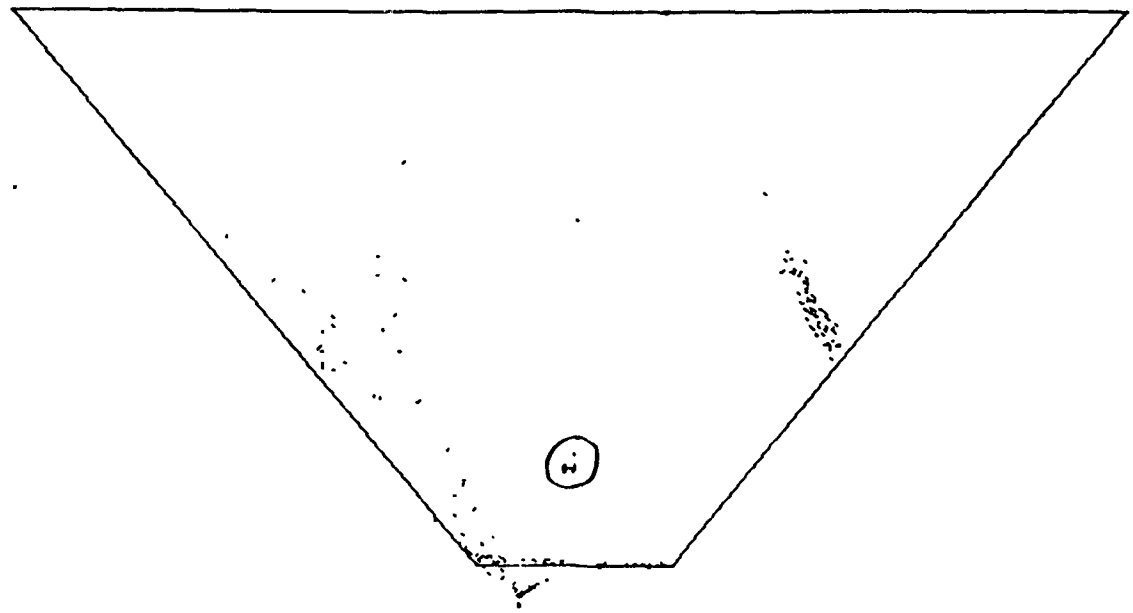


Figure 4.9a: Projection into Ground Plane of Obstacle Pixels with Mixed Pixel Minimization: Box at 25 feet

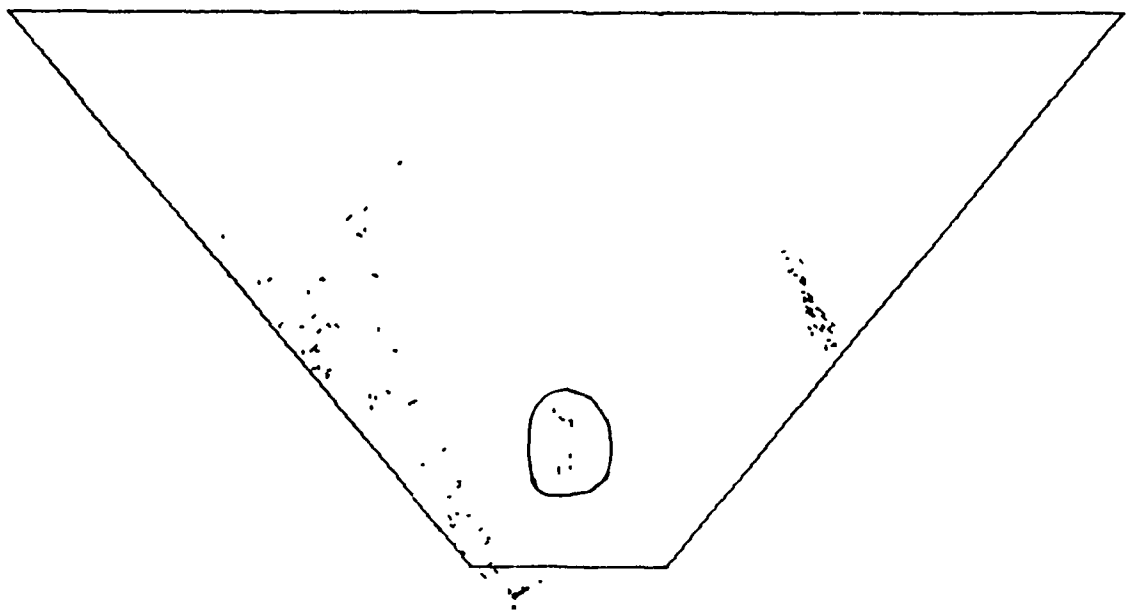
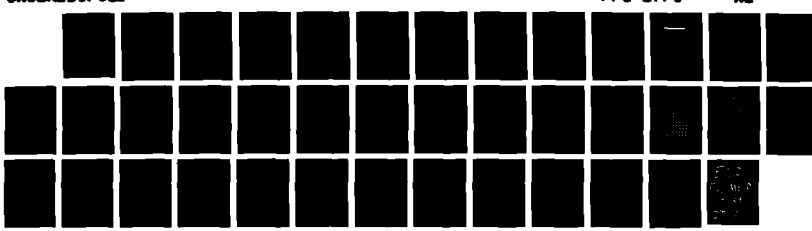
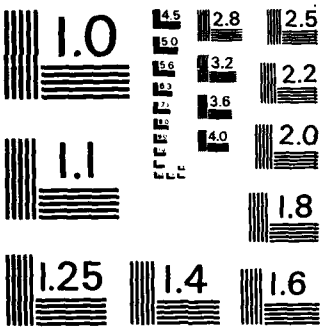


Figure 4.9b: Projection into Ground Plane of Obstacle Pixels without Mixed Pixel Minimization: Box at 25 feet

AD-A207 596 VISION-BASED NAVIGATION FOR AUTONOMOUS GROUND VEHICLES 2/2
(U) MARYLAND UNIV COLLEGE PARK CENTER FOR AUTOMATION
RESEARCH L S DAVIS NOV 88 ETL-0519 DACA76-84-C-0004
UNCLASSIFIED F/G 17/5 NL





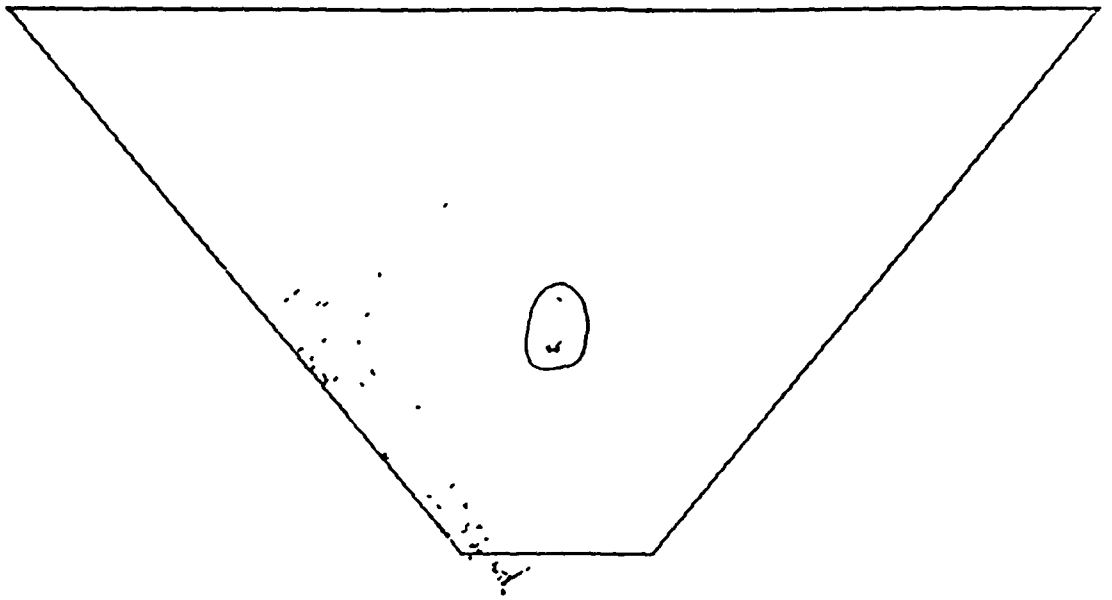


Figure 4.10a: Projection into Ground Plane of Obstacle Pixels with Mixed Pixel Minimization: Box at 40 feet

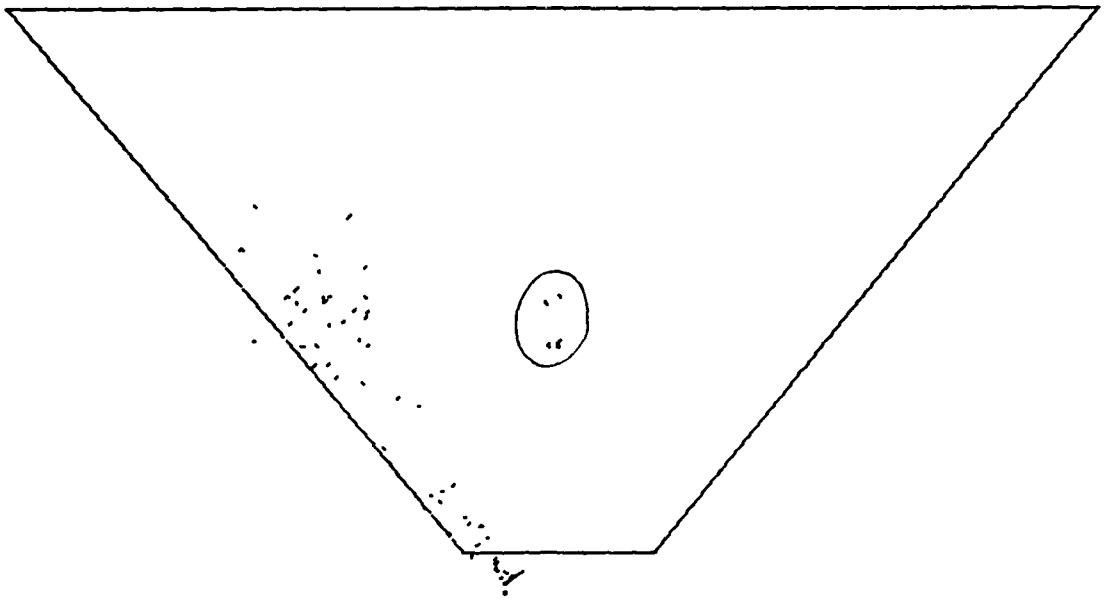


Figure 4.10b: Projection into Ground Plane of Obstacle Pixels without Mixed Pixel Minimization: Box at 40 feet

4.5. Conclusions

We have shown how range derivatives can be used for fast, reliable obstacle detection by an autonomous vehicle. The range derivative algorithm works well for relatively flat, on-road scenes as was shown in our computational experiments on both synthetic and real range images. It has significantly better performance, when the range scanner is perturbed, than other fast obstacle detection methods. For hilly terrain it will be necessary to use surface normals or the full $\Delta\rho/\rho$ term in place of the simpler $\Delta\rho$ approximation that was used in this report. We plan to study extensions to cross country navigation.

References

[1] T.B. Boult and J.R. Kender, "On Surface Reconstruction Using Sparse Depth Data," *Proceedings: Image Understanding Workshop*, Miami Beach, Florida, December 1985, 197–208.

[2] R.A. Brooks, "A Robust Layered Control System for a Mobile Robot," Artificial Intelligence Laboratory, A.I. Memo 864, Massachusetts Institute of Technology, Cambridge, MA, September 1985.

[3] B. Carrihill and R. Hummel, "Experiments with the Intensity Ratio Depth Sensor," *Computer Vision, Graphics and Image Processing*, 32 (1985), 337–358.

[4] D.J. Choi and J.R. Kender, "Solving the Depth Interpolation Problem with the Adaptive Chebyshev Acceleration Method on a Parallel Computer," *Proceedings: Image Understanding Workshop*, Miami Beach, Florida, December 1985, 219–223.

[5] D. DeMenthon, T.K. Siddalingaiah and L.S. Davis, "Production of Dense Range Images with the CVL Light-Stripe Range Scanner," Center for Automation Research Technical Report 337, University of Maryland, College Park, Maryland, December 1987.

[6] R.O. Duda, D. Nitzan and P. Barret, "Use of Range and Reflectance Data to Find Planar Surface Regions," *IEEE Transactions on PAMI*, 3 (1979), 259–271.

[7] R.N. Haber and M. Hershenson, *The Psychology of Visual Perception*, Holt, Reinhart and Winston, Inc., 1980.

[8] E.L. Hall, J.B.K. Tio, C.A. McPherson and F.A. Sadjadi, "Measuring Curved Surfaces for Robot Vision," *Computer* 15, 12 (1982) 42–54.

[9] D. Harwood, M. Subbarao, H. Hakalahti and L.S. Davis, "A New Class of Edge-Preserving Smoothing Filters," Center for Automation Research Technical Report 59, University of Maryland, College Park, MD, May 1984.

[10] M. Hebert and J. Ponce "A New Method for Segmenting 3-D Scenes into Primitives," *Proceedings: 6th International Conference on Pattern Recognition*, Munich, Germany, October 1982, 836–838.

[11] M. Hebert and T. Kanade, "First Results on Outdoor Scene Analysis Using Range Data," *Proceedings: Image Understanding Workshop*, Miami Beach, Florida, December 1985, 224–231.

[12] R. Hoffman and A.K. Jain, "Segmentation and Classification of Range Images", *Proceedings: Computer Vision and Pattern Recognition*, Miami Beach, Florida, December 1986, 424–426.

[13] R. Hoffman and A.K. Jain, "Segmentation and Classification of Range Images," Technical Report MSU-ENGR-86-002, Michigan State University, East Lansing, MI, 1986.

[14] S. Inokuchi, T. Nita, F. Matsuda and Y. Sakurai, "A Three Dimensional Edge-Region Operator for Range Pictures," *Proceedings: 6th International Conference on Pattern Recognition*, Munich, Germany, October 1982, 918–920.

[15] A.K. Jain and R. Hoffman, "Evidence Based Recognition of 3-D Object," MSU-ENGR-86-0013, Michigan State University, East Lansing, MI, 1986.

[16] R.A. Jarvis, "A Laser Time-of-Flight Range Scanner for Robotic Vision," *IEEE Transactions on PAMI 5*, 5 (1983), 505–512.

[17] R.A. Jarvis, "A Perspective on Range Finding Techniques for Computer Vision," *IEEE Transactions on PAMI 5*, 2 (1983) 122–139.

[18] S. Kambhampati and L.S. Davis, "Multiresolution Path Planning for Mobile Robots" *IEEE Journal of Robotics and Automation RA-2* (1986), 135–145.

[19] V. Larrowe, "Operating Principles of Laser Ranging, Image Producing (3D) Sensors," *Environmental Research Institute of Michigan*, Ann Arbor, MI, March, 1986.

[20] J. LeMoigne and A.M. Waxman, "Projected Grids for Short Range Navigation of Autonomous Robots," *Proceedings: 7th International Conference on Pattern Recognition*, Montreal, Canada, July 1984, 203–206.

[21] X.Y. Lin and W.G. Wee, "SDFS: A New Strategy for the Recognition of Objects Using Range Data," *Proceedings: IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 1986, 770–775.

[22] D. Nitzan, A.E. Brain and R.O. Duda, "The Measurement and Use of Registered Reflectance and Range Data in Scene Analysis," *Proceedings: IEEE 65*, (1977), 206–220.

[23] J.L. Olivier and F. Ozguner, "A Navigation Algorithm for an Intelligent Vehicle with a Laser Range Finder," *Proceedings: IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 1986.

[24] A.M. Parodi, J.J. Nitao and L.S. McTamaney, "An Intelligent System for an Autonomous Vehicle," *Proceedings: IEEE International Conference on Robotics and Automation* San Francisco, CA, April 1986, 1657–1663.

[25] B. Parvin and G. Medioni, "Segmentation of Range Images into Planar Surfaces by Split and Merge," *Proceedings: Computer Vision and Pattern Recognition*, Miami Beach, Florida, 1986, 415–417.

[26] D.W. Payton, "An Architecture for Reflexive Autonomous Vehicle Control," *Proceedings: IEEE International Conference on Robotics and Automation*, San Francisco, CA April 1986, 1838–1845.

[27] S. Puri and L.S. Davis, "Two Dimensional Path Planning with Obstacles and Shadows," Center for Automation Research Technical Report 225, University of Maryland, College Park, Maryland, January, 1987.

[28] R.R. Rogers, *A Short Course in Cloud Physics*, Pergamon Press, New York, 1979.

[29] J.T. Schwartz, "Structured Light Sensors for 3D Robot Vision," Courant Institute Robotics Research Report No. 8, New York University, New York, NY, 1983.

[30] I.K. Sethi and S.N. Jayaramamurthy, "Surface Classification Using Characteristic Contours," *Proceedings: 7th International Conference on Pattern Recognition*, Montreal, Canada, July 1984, 438–440.

[31] U.K. Sharma and L.S. Davis, "Road Following by an Autonomous Vehicle Using Range Data," Center for Automation Research Technical Report 194, University of Maryland, College Park, Maryland, March 1986.

[32] S.S. Sinha and A.M. Waxman, "An Image Flow Simulator," Center for Automation Research Technical Report 71, University of Maryland, College Park, Maryland, July 1984.

[33] G.B. Thomas, Jr., *Calculus and Analytic Geometry*, Addison-Wesley, Reading, MA 1972.

[34] B.C. Vemuri and J.K. Aggarwal, "3-Dimensional Reconstruction of Object from Range Data," *Proceedings: 7th International Conference on Pattern Recognition*, Montreal, Canada, July 1984, 752–754.

[35] A.M. Waxman, J. LeMoigne, L.S. Davis, E. Liang and T. Siddalingaiah, "A Visual Navigation System for Autonomous Land Vehicles," Center for Automation Research Technical Report 139, University of Maryland, College Park, MD July 1985.

[36] H.S. Yang, K.L. Boyer and A.C. Kak, "Range Data Extraction and Interpretation by Structured Light," *Proceedings: Conference on Artificial Intelligence Applications*, Denver, Colorado, December 1984, 199–205.

[37] D.M. Zuk and M.L. Dell'Eva, "Three-Dimensional Vision System for the Adaptive Suspension Vehicle," ERIM Report 170400-3-f, Ann Arbor, Michigan, January 1983.

5. ROAD STRUCTURE FROM MOTION

In this section, we describe a new method for reconstructing the 3-D structure of road boundaries from consecutive images. First, we present a method for estimating depth information; this is a motion stereo method applied to consecutive images, given an estimate of the interframe motion. The relation between depth, motion and disparity is investigated, since the accuracy of the depth depends on the disparity range. Next, the error of the estimated road structure due to quantization errors and motion estimation error is examined. Finally, a representation for road boundaries is proposed that makes explicit the error of the road edge location in 3-D space. Experimental results are shown for an input image sequence taken from a scale model of a road scene.

5.1. The Principle of Motion Stereo Used in this Method

Our motion stereo analysis is based on the following assumptions.

(1) The road boundaries in consecutive images are detected and the correspondence between frames is established.

(2) The robot motion is known.

Algorithms [Waxman et al.] have been developed for road boundary detection in images. Our program uses the output of one such algorithm. Also, given the interframe motion, determining the correspondence between road boundaries in consecutive frames is straightforward. Internal sensors can be employed for accurately estimating the motion of the robot.

We outline the depth recovery algorithm as follows. Given the interframe motion, the epipolar line in the second image corresponding to any point P in the first image can be determined by projecting the epipolar plane determined by P and the two camera centers onto the second image [Echigo and Yachida]. We can, however, determine this epipolar line more directly. The known motion consists of a translation $T=(U,V,W)$ and a rotation $R=(\alpha,\beta,\gamma)$. The displacement vector of a feature point in the first frame consists of two vectors t (translation) and r (rotation) in the image plane. While r is uniquely determined when R is known, t has one degree of freedom—length. The direction of t is constrained so as to be toward the focus of expansion (the intersection of T with the image plane [9]). Thus, we can determine the epipolar line as follows:

(1) Transform the feature point P_1 in the first frame by the rotation r. The displaced location is denoted by P_1' (see Figure 5.1).

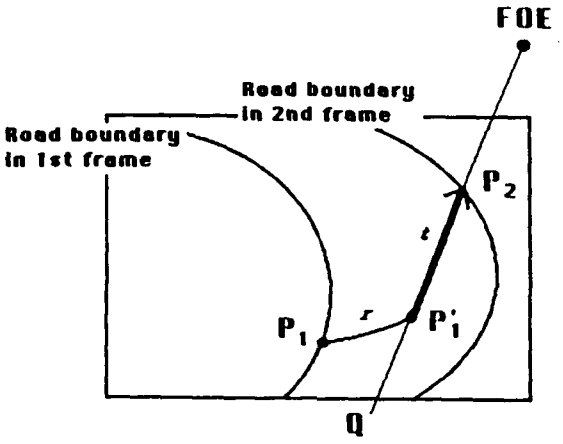


Figure 5.1: Epipolar Line in the Motion Stereo

(2) Draw a line from P_1' to the FOE.

(3) The line segment P_1'-FOE is the desired epipolar line. The corresponding point in the second frame must lie on this line segment (from P_1' to FOE) when $W<0$ or on the half line from P_1' to Q (see Figure 5.1) when $W>0$.

This method for constructing the epipolar line does not require calculation of the equation of the epipolar plane in 3-D space. Furthermore, the constraint (3) is not explicit in previous methods.

The translation vector $t=(d_x,d_y)$ is easily determined by finding the intersection between the epipolar line segment and the road boundary in the second frame (see Figure 5.1). The 3-D coordinates of the feature point can be recovered from the translation vector (d_x,d_y). For simplicity, we assume that the first frame has already been corrected for the known rotation R; therefore, only the translation (U,V,W) need be considered. The camera translation (U,V,W) can be interpreted as a translation $(-U,-V,-W)$ of the feature point with respect to a fixed camera. Therefore, the coordinates of the feature point (x_1,y_1) in the first frame and (x_2,y_2) in the second frame are given as:

$$x_1=\frac{fX}{Z},\ y_1=\frac{fY}{Z},\ x_2=\frac{f(X-U)}{Z-W},\ \text{and}\ \ y_2=\frac{f(Y-V)}{Z-W} \qquad (1.1)$$

where f denotes the focal length of the camera and (X,Y,Z) represent the 3-D coordinate of the point in the first camera coordinate system. From eqns(1.1)

$$d_x = x_2 - x_1 = -\frac{fU}{Z-W} + \frac{fXW}{Z(Z-W)}$$

and (1.2)

$$d_y = y_2 - y_1 = -\frac{fV}{Z-W} + \frac{fYW}{Z(Z-W)}$$

Finally, we obtain the 3-D coordinates:

$$Z = -\frac{fU}{d_x} + W\left(\frac{x_1}{d_x} + 1\right) \text{ or } Z = -\frac{fV}{d_y} + W\left(\frac{y_1}{d_y} + 1\right) \quad (1.3)$$

and

$$X = \frac{x_1 Z}{f}, \qquad Y = \frac{y_1 Z}{f},$$

The recovery of road structure from motion stereo does not, of course, require prior knowledge of camera height, tilt angle or road width as do most monocular inverse perspective methods.

5.2. Properties of the Disparity for a Mobile Robot

We consider how the disparity relates to the tilt angle and the motion. Figure 5.2 shows the parameters of our computer simulation. The image size is 512×480. The visual angle of the camera is approximately 33 degrees, corresponding to about 800 pixels of equivalent focal length. The tilt angle of the camera is approximately 23 degrees. The camera height is approximately 50 mm; we use this length as unit length in the simulations. In the following, we set $U=V=0$ to simplify the analysis; these would be zero if the robot moved straight on a flat surface.

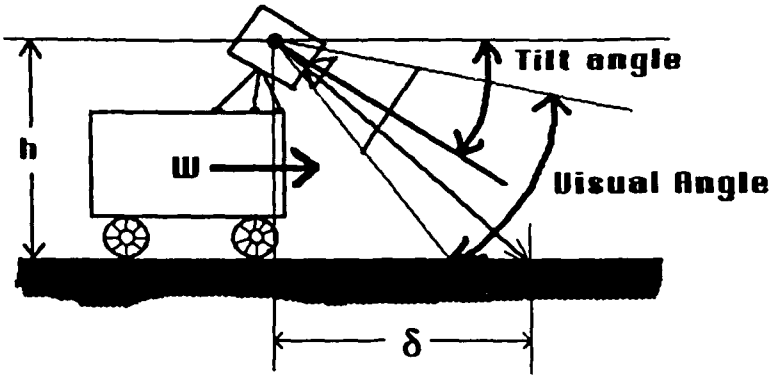


Figure 5.2: Geometrical Parameters of ALV

(a) **Disparity vs. Tilt Angle**

When $U=V=0$, the disparity (d_x, d_y) is given by

$$d_x=\frac{fXW}{Z(Z-W)}=\frac{x_1W}{Z-W},\quad d_y=\frac{fYW}{Z(Z-W)}=\frac{y_1W}{Z-W}.$$

Here we need consider only the disparity d_y because the tilt angle does not affect X and U. If we assume a flat surface and no tilt ($\theta=0$), $Y=h$ (camera height); therefore

$$d_y=\frac{fhW}{Z(Z-W)}. \tag{2.1}$$

If the camera has a tilt angle θ, then the camera has an apparent translation component in the direction of the y-axis, although the robot has no translation component in the vertical direction ($V=0$); therefore, eqn(2.1) becomes

$$d_y=\frac{fhW}{(h\sin\theta+Z\cos\theta)(h\sin\theta+Z\cos\theta-W\cos\theta)}. \tag{2.2}$$

(In the following, (U,V,W) refers to the robot translation. Figure 5.3 displays the relationship between disparity and tilt angle for the curve when $W=0.1$

(h=1: unit length). The horizontal axis corresponds to the distance δ between the camera and the ground, and the ver ical axis to disparity d_y. The bar graphs below these curves represent the extent of the visual fields, which depend on both θ and the vertical field of view of the camera. Based on these graphs, we can make the following observations.

1) For a fixed δ, the tilt angle does not have much effect on d_y.

2) As θ increases, the visual field is compressed towards the camera, and near the camera, where disparity d_y is large, accurate depth information can be obtained.

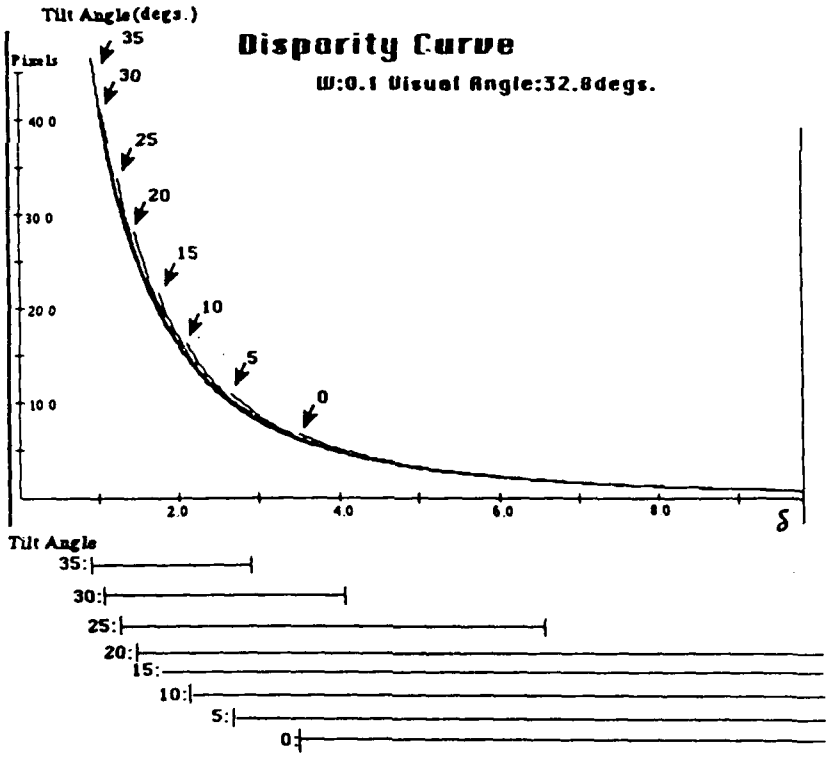


Figure 5.3: Disparity Curves vs. Tilt Angles

In the following, the tilt angle is fixed to 23 degrees: this scales the vertical visual field from 1.4 to 8.7 units in front of the camera when $W=0.1$.

(b) **Disparity vs. Motion**

In an ordinary stereo vision system, where the camera's lines of sight are parallel and the base line is perpendicular to the line of sight, disparity d_x is proportional to the length of the base line (U):

$$d_x = -\frac{fU}{Z} \qquad (V=W=0). \tag{2.3}$$

Next, we examine the relationship between disparity and motion W. From (2.2), the relationship between disparity and motion when $W=w$ is

$$d_y^w = \frac{fhw}{Z'(Z'-w\cos\theta)}$$

where $Z'=h\sin\theta+Z\cos\theta$. If we increase the motion w by a factor of n ($W=nw$)

$$d_y^{nw} = \frac{fhnw}{Z'(Z'-nw\cos\theta)}.$$

If F is the ratio of disparity for velocity nw to disparity at velocity w, then

$$F = \frac{d_y^{nw}}{d_y^w} = n + \frac{n(n-1)w\cos\theta}{Z'-nw\cos\theta}. \tag{2.4}$$

Figure 5.4 shows curves of F in terms of n with $w=0.1$. We can make the following observations.

1) From this graph and (2.4), $F \approx n$ when $w \ll Z'$.

2) In an ordinary stereo system, the common visual field of the two cameras changes from 70% to 30% when the base line changes from $U=0.1$ to

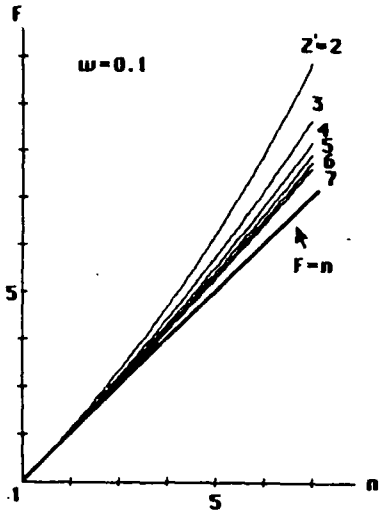


Figure 5.4: Factor Curve in Terms of Motion Parameters

U=0.2. In motion stereo, the common visual field of two frames changes only from 95% to 84% when the motion changes from W=0.1 to W=0.2.

5.3. Error Analysis of Estimated Depth in 3-D Space

The estimated depth information has errors due both to quantization errors of edge locations and to estimation errors of robot motion. In this section, we consider the errors in the depth information in 3-D space.

5.3.1. Depth Error Due to Quantization Error in Edge Location

Since the disparity is determined from the difference of the corresponding feature points in two frames, inaccuracy of the disparity results from localization errors of feature points in the image plane. If a point has a quantization error e_{yi} in the i^{th} frame (i=1,2), the disparity has a total quantization error e_y given by

$$| e_y | = | e_{y1} | + | e_{y2} | .$$

Figure 5.5 shows a side view of the depth error due to quantization error of edge location. An actual point must exist in a rectangle $PQRS$ (a solid in 3-D space) which is obtained as a common region of two thick rays from the first and the second camera positions. Since PR is very small compared to the length QS, we approximate this rectangle as a "stick" QS in 3-D space.

An important property of the depth errors due to quantization errors is that they are independent of one another.

5.3.2. Relative Depth Errors

We can estimate the relative errors in the depth information by our method as follows.

From (1.3),

$$Z = -\frac{fV}{d_y} + \frac{Wy_2}{d_y} = \frac{1}{d_y}(-fV + Wy_2).$$

If the disparity error is uniformly distributed in the interval $[-e_y, +e_y]$, then the maximal overestimated depth Z_o and the minimal underestimated depth Z_u are

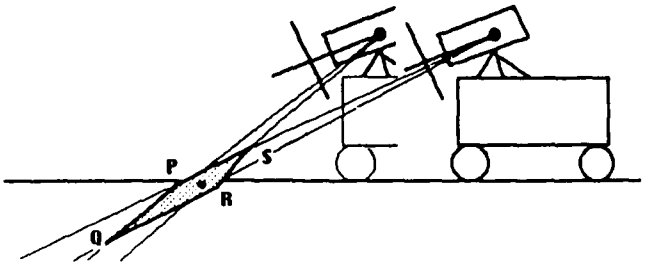


Figure 5.5: Side View of Depth Errors Due to Quantization Error of Edge Location

given by:

$$Z_o = \frac{1}{d_y - e_y}(-fV + Wy_2) \quad \text{and} \quad Z_u = \frac{1}{d_y + e_y}(-fV + Wy_2).$$

ΔZ, the expected error in Z, is thus:

$$\Delta Z = \frac{Z_o - Z_u}{2} = \frac{e_y}{d_y^2 - e_y^2}(-fV + Wy_2).$$

Therefore, the relative error $\left|\frac{\Delta Z}{Z}\right|$ is given by:

$$\left|\frac{\Delta Z}{Z}\right| = \frac{d_y e_y}{d_y^2 - e_y^2}.$$

When $e_y \ll d_y$, $\left|\frac{\Delta Z}{Z}\right| \approx \frac{e_y}{d_y}$: the relative error of the depth value is inversely proportional to the disparity value.

5.3.2.1. Results of Simulation

We applied our method to synthetic road boundaries. In the following simulation, we used the same camera parameters as in above (tilt angle is fixed at 23 degrees). Figure 5.6(a) shows the road boundaries extracted from first frame (+) and second frame (○) with translation (U,V,W)=(0,0,0.4) and rotation (α,β,γ)=(0,-5.0°,0) (steering to the right). The locations of feature points on the road boundary are quantized in a 512×480 pixel image. Figure 5.6(b) shows the 3-D road structure in the world coordinate system from the top and side. Large circles in the top view show the locations of the robot in the first frame (+) and in the second frame (small ○). Figure 5.7 shows the results of reconstruction.

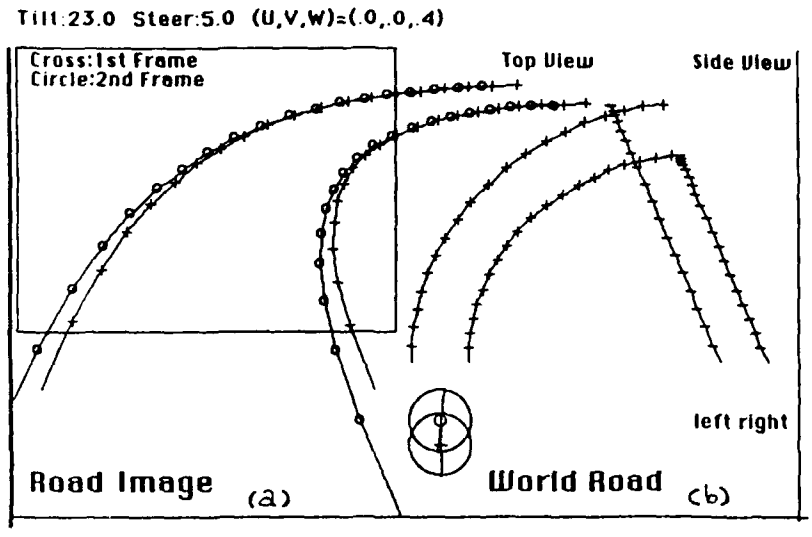


Figure 5.6: Road Boundaries with Translation and Rotation

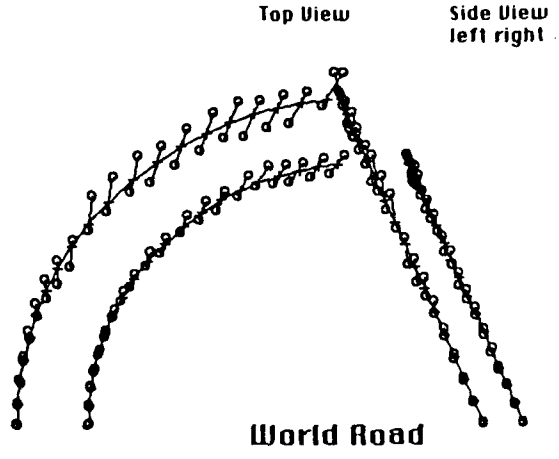


Figure 5.7: Simulation Result for 5.6

Short line segments (which we will refer to as "sticks") with small circles at both ends represent intervals of edge location in 3-D when e_y=1 pixel. The true location (+) lies on the corresponding stick. The nearer the point is to the viewer, the shorter is the length of the stick. The relative depth error was very nearly inversely proportional to disparity as mentioned above.

5.3.3. Discussion

We have described a new method for reconstructing the depth information of a road boundary from motion stereo and for representing the error of a 3-D road structure in 3-D space. The depth errors due to inaccuracy of motion estimation are especially important because the errors in the translational components change the depth values of all points in the same way. In this section, we have not considered the effect of the rotational movement on the depth value. A rotational motion about the X- or Y-axis of 1° causes a motion of 10 pixels in the image plane if the same focal length lens is used (about 800 pixels of equivalent focal length). Displacement due to a rotation around the Z-axis depends on the distance from the image center and the maximum displacement is 4 or 5 pixels at the edge of the image. Therefore, an error in rotation estimation can cause serious errors in disparity. The final goal of our research is to obtain a more accurate 3-D road structure by fusing the error ranges of depth information from many more frames and/or from range sensors in 3-D space.

References

[1] B.K.P. Horn, *Robot Vision*, MIT Press, Cambridge, Massachusetts, 1986.

[2] S. Tsuji, Y. Yagi and M. Asada, "Dynamic Scene Analysis for a Mobile Robot in Man-Made Environment", *IEEE Int. Conf. Robotics and Automation*, pp. 850–855, 1985.

[3] S. Tsuji, J. Y. Zheng and M. Asada, "Stereo Vision of a Mobile Robot: World Constraints for Image Matching and Interpretation", *IEEE Int. Conf. Robotics and Automation*, pp. 1594–1599, 1986.

[4] D. DeMenthon, "A Zero-Bank Algorithm for Inverse Perspective of a Road from a Single View", *IEEE Int. Conf. Robotics and Automation*, pp. 1444–1449, 1987.

[5] A.M. Waxman, J. LeMoigne, L.S. Davis, E. Liang and T. Siddalingaiah, "A Visual Navigation System", *IEEE Int. Conf. Robotics and Automation*, pp.

1600–1606, 1986.

[6] M. Asada and S. Tsuji, "Shape from Projecting a Stripe Pattern", *IEEE Int. Conf. Robotics and Automation*, pp. 787–792, 1987.

[7] L.S. Davis, T. Kushner, J. LeMoigne and A. Waxman, "Road Boundary Detection for Autonomous Vehicle Navigation", *Optical Engineering*, **25**, pp. 409–414, 1986.

[8] T. Echigo and M. Yachida, "A Fast Method for Extracting of 3-D Information Using Multiple Stripes and Two Cameras", *9th IJCAI*, pp. 1127–1130, 1985.

[9] K. Prazdny, "Determining the Instantaneous Direction of Motion from Optical Flow Generated by a Curvilinearly Moving Observer", *Computer Graphics and Image Processing*, **17**, pp. 238–248, 1981.

[10] G. Xu, S. Tsuji and M. Asada, "A Motion Stereo Method Based Coarse-to-Fine Control Strategy", *IEEE Trans. PAMI* **PAMI-9**, pp. 332–336, 1987.

[11] R.Y. Tsai, "An Efficient Accurate Camera Calibration for 3-D Machine Vision", CVPR86, pp. 364–374, 1986.

[12] American Robot Corporation, "Merlin Robot System Operator and User Guide", Manual No. SMT-2.0-0184, revision 2.1, 1984.

6. FAT PYRAMIDS AND CONNECTION MACHINE VISION

6.1. Introduction

This section is concerned with an algorithm for the log n time computation of the complete histogram of an $n \times n$ gray-level array. It makes use of a "fat" pyramid implemented on an SIMD hypercube multiprocessor with very high processor utilization. These terms are explained below.

[Tanimoto] describes an algorithm for histogram computation on a standard pyramid which involves performing one log n time "pass" from the bottom of the pyramid to the top for each gray-level value to be computed. The algorithm presented below for Fat Pyramids computes the complete histogram in a single pass.

The algorithm is perhaps more properly called a hypercube multiprocessor algorithm. However, the abstraction of the Fat Pyramid provides a convenient way of describing the computations involved.

6.2. SIMD Hypercube Multiprocessors

A hypercube multiprocessor consists of 2^M (for some M) processing cells interconnected as if each were located at one vertex of an M-dimensional hypercube, so that any two cells share a direct connection if and only if their corresponding hypercube vertices are connected by a hypercube edge. Furthermore, if the hypercube has unit side, and we assign to each cell an address given by the M-dimensional coordinates of the corresponding vertex, then we can see that two cells will share a direct connection if and only if their addresses differ in

exactly one bit position.

In a SIMD (Single-Instruction stream, Multiple-Data stream) multiprocessor, all processing cells execute in unison a stream of commands broadcast from a single controller. The exception to this is that some subset of the cells may ignore a given instruction; the decision to do so being based on the contents of their local cell memory.

6.3. Fat Pyramids

A grid multiprocessor consists of a rectangular array of nodes, with a processing cell at each node, each of which is directly connected to its four nearest neighbors (except on the edges of the grid). In a pyramid multiprocessor, there are several such levels of grids, each containing (say) one-fourth as many nodes as the one below, and with each node connected directly to four nodes in the level below. This arrangement allows for some combination of the information in all of the nodes at the bottom level to be computed in log n time, where the bottom level is (say) an $n \times n$ square, by combining at each node in level $l+1$ the information from that node's four children at level l (level 0 is the bottom level).

In a 'fat' pyramid, the size of a processor associated with a node in the pyramid depends on the level of the pyramid in which it appears. Specifically, a processor at level $l+1$ will have four times as much storage, and possibly four times as much processing power, as any processor at level l. A Fat Pyramid allows for combining operations in which the amount of information per node increases with the level of the node in the pyramid.

6.4. Fat Pyramids and Hypercubes

The pyramid algorithm presented below has the property that only one level is 'active' at any given time; that is, while a computation is being carried out on a given level, no other level needs to perform a computation. Other pyramid algorithms also have this property. In such cases, it does not seem desirable to have processors lying idle when the level with which they are associated is not active. Suppose instead that some multiprocessor system were multiplexed among all the levels, with all of its processing cells being divided up among the nodes of a given level when that level is active. Thus if one cell were associated with each node at level 0 of the pyramid, there would be four cells cooperating to carry out the computations for each node at level 1, sixteen cells grouped to perform the operations required by each node at level 2, and in general 4^l cells associated with each pyramid node at level l. I will call such an arrangement a 'collapsed' Fat Pyramid.

In order for several cells to cooperate efficiently in performing some computation, it would be useful if there were a high degree of interconnectivity among them. Just such arrangement can be nicely provided on a hypercube. At any level l of the pyramid, all of the cells of the hypercube are divided up among the nodes at that level, in blocks of 4^l cells per node. It is possible to do this so that the block of cells implementing each node are themselves connected in a hypercube of dimension $2l$; this way none of the 4^l cells in a block are more than $2l$ links apart. Also, if we define two processor blocks to be adjacent if any cell in one block is directly connected to some cell in the other block, then we can also

arrange for all of the blocks at any given level to form a rectangular grid by this kind of adjacency, which is of course what we desire. Further discussion of such an arrangement follows.

6.5. Gray Codes and Grids

[Chan and Saad] describe how to embed a grid in a hypercube by the use of *Gray Codes*, which are number sequences in which the binary representations of two adjacent elements differ in exactly one bit position. The Gray Code used in what follows will be the *binary reflected Gray Code*, for which the i-bit sequence is obtained by appending a reversed copy of the i–1-bit sequence to itself with each element prefixed by a 1. (So for example, the 2-bit binary reflected Gray Code is {00, 01, 11, 10}, and the 3-bit sequence is {000, 001, 011, 010, 110, 111, 101, 100}.)

Consider a cell with grid coordinates (x,y), each of which have p bits, and let

$$g_x = g_{x,p-1}g_{x,p-2}\cdots g_{x,0}$$

and

$$g_y = g_{y,p-1}g_{y,p-2}\cdots g_{y,0}$$

be the x^{th} and y^{th} elements of the p-bit binary reflected Gray Code, respectively. Then the hypercube address of this cell is taken as the concatenation

$$g_{x,p-1}g_{x,p-2}\cdots g_{x,0}g_{y,p-1}g_{y,p-2}\cdots g_{y,0}.$$

It is not hard to see that the hypercube addresses of two cells adjacent in the

grid will have addresses differing in exactly one bit position. Thus any two such cells will also be adjacent in the hypercube. This mapping is illustrated in Figure 6.1.

6.6. A New Mapping

If we instead *interleave* the bits of g_x and g_y, giving

$$g_{x,p-1}g_{y,p-1}g_{x,p-2}g_{y,p-2}\cdots g_{x,0}g_{y,0},$$

then we get not only a mapping from a hypercube to a grid which preserves adjacency, but we also get a natural hypercube-to-collapsed-fat-pyramid mapping in the following sense. This mapping allows the hypercube address of any cell in

7	4	12	28	20	52	60	44	36
6	5	13	28	21	53	61	45	37
5	7	15	31	23	55	63	47	39
4	6	14	30	22	54	62	46	38
3	2	10	26	18	50	58	42	34
2	3	11	27	19	51	59	43	35
1	1	9	25	17	49	57	41	33
0	0	8	24	16	48	56	40	32
	0	1	2	3	4	5	6	7

Figure 6.1: Hypercube to Grid Mapping

any block at any level l to be regarded as the concatenation of two binary numbers, the first one being a $2(L-l)$ bit number (where L is the number of levels in the pyramid) determining which processor block the cell is in, and the other being a $2l$ bit number giving a local address of the cell within its block. In other words, all of the cells comprising any block on any level l will have hypercube addresses which are all in the range $k4^l$ to $(k+1)4^l-1$ for some k. This is illustrated in Figures 6.2 through 6.4. (Notice that this property does *not* hold for the mapping depicted in Figure 6.1.)

7	16	18	26	24	56	58	50	48
6	17	19	27	25	57	59	51	49
5	21	23	31	29	61	63	55	53
4	20	22	30	28	60	62	54	52
3	4	6	14	12	44	46	38	36
2	5	7	15	13	45	47	39	37
1	1	3	11	9	41	43	35	33
0	0	2	10	8	40	42	34	32
	0	1	2	3	4	5	6	7

Figure 6.2: Hypercube to Fat Pyramid Mapping
Division of Cells Among Nodes at Pyramid Level 0

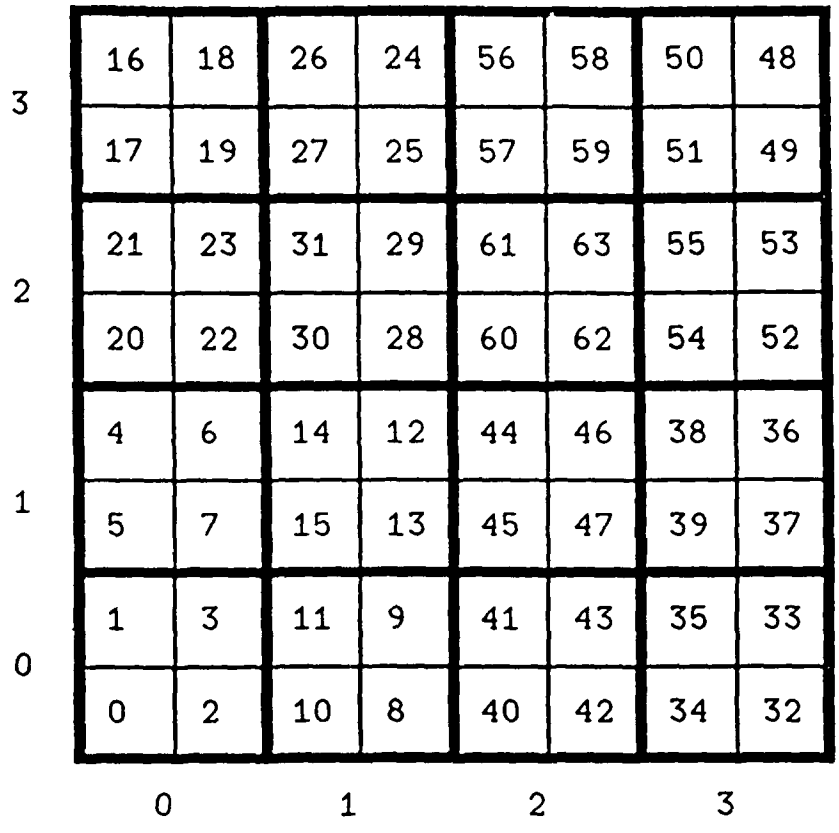


Figure 6.3: Hypercube to Fat Pyramid Mapping
Division of Cells Among Nodes at Pyramid Level 1

Consider, for example, cell 57 in Figure 6.3, which depicts a four-level collapsed fat pyramid at level 1. Now, 57 in binary is 111001, so this is cell 01 within processor block 1110. Notice also that if we apply in reverse the mapping described above to just the number 1110, we get (2,3) (de-interleaving the bits of 1110 gives 11 and 10, and these are the two-bit binary reflected Gray Code values g_2 and g_3, respectively), which is precisely the position within the level 1 grid of the block containing cell 57.

The property that a cell's hypercube address is also simply the concatenation of its block number and local address is not essential to the algorithm presented

below, but greatly simplifies its expression.

6.7. A Sketch of the Histogram Algorithm

Given an nxn array of gray-level values, the problem is to compute a histogram of these values over the array. Suppose that each gray-level value is stored in one cell at the bottom level of a Fat Pyramid implemented on a hypercube as described above.

Essentially, each block of processors at a given level will contain the histogram for the gray-level values of the cells in that block. Processing proceeds

	16	18	26	24	56	58	50	48
	17	19	27	25	57	59	51	49
1	21	23	31	29	61	63	55	53
	20	22	30	28	60	62	54	52
	4	6	14	12	44	46	38	36
	5	7	15	13	45	47	39	37
0	1	3	11	9	41	43	35	33
	0	2	10	8	40	42	34	32
		0				1		

Figure 6.4: Hypercube to Fat Pyramid Mapping
Division of Cells Among Nodes at Pyramid Level 2

from one level to the next by combining the histograms of the four child blocks to make up the histogram of the parent block. (Although this seems similar to a divide-and-conquer algorithm on a grid, the two schemes are different in that the cells constituting each block in any level of the collapsed Fat Pyramid are interconnected to form a hypercube. Furthermore, there is greater interconnectivity between adjacent blocks in the collapsed Fat Pyramid.)

Suppose that the gray-level values are s bits long (assume s even). The algorithm consists of two different phases, one for levels below $s/2$, and one for the higher levels. This is because at levels below $s/2$, the width of a cell's local address is less than the width of a gray-level value. The significance of this fact will become clear below.

Let the current level be $l < s/2$. Then the algorithm works as follows. (Assume in what follows that k always represents $L-l$.) Consider four blocks B_0, B_1, B_2, and B_3 at level l, which are to be combined into a larger block B at level $l+1$. Consider a cell in any B_i with local address c. Let its complete address be $b_{2k-1}b_{2k-2}\cdots b_0c_{2l-1}c_{2l-2}\cdots c_0$. We assume at this point in the algorithm that this cell contains the histogram information over B_i for all gray-level values of the form

$$g_{s-1}g_{s-2}\cdots g_{2l}c_{2l-1}c_{2l-2}\cdots c_0,$$

that is, ending in c. (Efficient schemes for doing this are disussed below. It turns out that for a pyramid with 2^{20} nodes on the bottom level, each containing an 8-bit gray-level value, no level of the pyramid requires more than 36 bits per

cell for the storage of all necessary histogram information.) Once the blocks are merged to form B, we will want any cell with address

$$b_{2k-1}b_{2k-2}\cdots b_2c_{2l+1}c_{2l}c_{2l-1}c_{2l-2}\cdots c_0$$

to contain the histogram information over B for all gray-level values of the form

$$g_{s-1}g_{s-2}\cdots g_{2l+2}c_{2l+1}c_{2l}c_{2l-1}c_{2l-2}\cdots c_0.$$

Consider, for example, a cell in B_0, with local address c. Let its complete address be

$$b_{2k-1}b_{2k-2}\cdots b_200c_{2l-1}c_{2l-2}\cdots c_0.$$

This cell must collect the histogram information over each B_i for all the gray-level values of the form

$$g_{s-1}g_{s-2}\cdots g_{2l+2}00c_{2l-1}c_{2l-2}\cdots c_0.$$

Fortunately, all of this information is distributed across the cells at

$$b_{2k-1}b_{2k-2}\cdots b_200c_{2l-1}c_{2l-2}\cdots c_0$$

which is the cell itself,

$$b_{2k-1}b_{2k-2}\cdots b_201c_{2l-1}c_{2l-2}\cdots c_0,$$

which is a cell in block B_1,

$$b_{2k-1}b_{2k-2}\cdots b_210c_{2l-1}c_{2l-2}\cdots c_0,$$

which is a cell in block B_2, and

$$b_{2k-1}b_{2k-2}\cdots b_211c_{2l-1}c_{2l-2}\cdots c_0,$$

which is a cell in block B_3. This is because no other cells in B have addresses ending in c, and so no other cells in B can contain histogram information over B for gray-level values ending in c. But we see by inspecting the addresses of the four cells given above that they are connected in a square by hypercube edges. So it will take only two parallel transmission operations to perform the necessary redistribution of the histogram information.

At level $l = s/2$, consider a cell in block B with local address c. This cell will contain the histogram information over B for all gray-level values ending in c. But c is $2l = s$ bits long. Thus the cell contains only the histogram information over B for the single gray-level value c.

For levels $l \geq s/2$, the algorithm works as follows. Within a block B at some level l, we want any cell whose address ends in

$$g = g_{s-1}g_{s-2}\cdots g_0$$

to contain the histogram information over B for the single gray-level value g. We can see that this is true when we reach level $s/2$. Now consider the four level l blocks B_0, B_1, B_2, and B_3. Let the addresses of these blocks be

$$b_{2k-1}b_{2k-2}\cdots b_200,$$

$$b_{2k-1}b_{2k-2}\cdots b_201,$$

$$b_{2k-1}b_{2k-2}\cdots b_210,$$

and

$$b_{2k-1}b_{2k-2}\cdots b_211,$$

respectively.

Suppose that these four blocks are to be combined to form the level $l+1$ block B. We must combine the histogram information from each of those blocks to form the histogram for B.

For example, the histogram information over B_0 for the gray-level value

$$g = g_{s-1}g_{s-2}\cdots g_0$$

is contained at any cell in B_0 with local address ending in g, that is, with complete address

$$b_{2k-1}b_{2k-2}\cdots b_2 00 c_{2l-1}c_{2l-2}\cdots c_s g_{s-1}g_{s-2}\cdots g_0$$

for some c_i's. But we know also that the histogram information over B_1, B_2, and B_3 for the gray-level value g can be found in

$$b_{2k-1}b_{2k-2}\cdots b_2 01 c_{2l-1}c_{2l-2}\cdots c_s g_{s-1}g_{s-2}\cdots g_0,$$

$$b_{2k-1}b_{2k-2}\cdots b_2 10 c_{2l-1}c_{2l-2}\cdots c_s g_{s-1}g_{s-2}\cdots g_0,$$

and

$$b_{2k-1}b_{2k-2}\cdots b_2 11 c_{2l-1}c_{2l-2}\cdots c_s g_{s-1}g_{s-2}\cdots g_0,$$

respectively. These four cells are connected in a square by hypercube edges. So in two parallel transmission operations the cell in each block can collect the histogram information from the cells in the other three blocks, and can compute the histogram information over the entire block B for the gray-level value g. The algorithm is presented in more detail in [Bestul and Davis].

6.8. Summary and Comments

This section has presented a $\log(n)$ time algorithm for the computation of the histogram of a collection of values distributed one per cell across a hypercube multiprocessor. It uses the technique of regarding the hypercube as a multiplexed implementation of a 'fat' pyramid, in which the processor at each pyramid node is larger than the processors at that node's children.

The algorithm requires at most $\sqrt{2^s}$ space per cell where s is the number of bits required to store one of the histogram domain values, although much less storage may be necessary if appropriate encodings of intermediate results are used. Thus it is most appropriate for use where s is less than $2L$, that is, when the possible range of the values is less than the total number of values. Machine vision applications involving arrays of tens or hundreds of thousands of gray-level values quantized to a few hundred or thousand possible levels are natural applications for this algorithm.

In some cases it is desirable for the resulting histogram array to be stored such that values adjacent in the histogram array are located in adjacent processors. Consider the case of a two-dimensional array of gray-scale values, and regard this array as a mapping from position to value. We see that the gray-scale values are the *dependent* values of this mapping. But notice that their role changes to that of *independent* values when considering the histogram itself as a mapping. We were able to have the (dependent) gray-scale values of adjacent (independent) positions reside in adjacent processors by first transforming the

coordinates for a given position to Gray Code values, interleaving (or concatenating) these transformed coordinates, and storing the corresponding gray-scale value in the processor whose address is given by the interleaved (or concatenated) Gray Code values.

In the computed histogram, the gray-scale values play the role of independent values of a mapping whose dependent values are indexed by processor addresses. This suggests by analogy how to obtain the desired adjacency of the (dependent) histogram values. Specifically, consider what happens if we transform each gray-scale value to its corresponding Gray Code value, before performing the histogram computation. Then, when the histogram computation is complete, the histogram value mapped to by some given gray-scale value can be found by converting that gray-scale value to a Gray Code value, and then accessing a cell whose address ends in the Gray Code value. Note that two sequential gray-scale values have corresponding Gray Code values differing in exactly one bit position, and so the histogram values mapped to by these two gray-scale values will be found in adjacent cells.

This idea of course generalizes to histograms with multi-dimensional domains, where it may be desired that the histogram values corresponding to adjacent values in the domain be found in adjacent cells. In this case, each of the elements in the histogram domain tuple in each cell is first converted to a Gray Code value. Then these Gray Code tuples are interleaved (or concatenated), and the histogram of the resulting values is computed. Now, in order to access the histogram value for a given domain tuple, the same transformation

originally applied to the domain tuples is applied to the given tuple, and the corresponding cell accessed. We see here that two adjacent domain tuples, that is, two tuples differing by one in one of their elements, will have converted (interleaved Gray Code) values differing in one bit position, and so will map to histogram values which are found in adjacent cells.

References

[1] T. Bestul and L.S. Davis, "On Computing Histograms of Images in log n Time Using Fat Pyramids," University of Maryland Center for Automation Research Technical Report 271, College Park, MD.

[2] T.G. Chan and Y. Saad, "Multigrid Algorithms on the Hypercube Multiprocessor," *IEEE Transactions on Computers*, vol. C-35, No. 11, pp. 969–977, November 1986.

[3] S.L. Tanimoto, "Sorting, Histogramming, and Other Statistical Operations on a Pyramid Machine," in *Multiresolution Image Processing and Analysis*, ed. A. Rosenfeld, pp. 136–145, Springer-Verlag.

7. REPORTS

Following is a list of the reports generated under this contract during the period July 1, 1986 through June 30, 1987.

[1] Daniel Dementhon, "Inverse Perspective of a Road from a Single Image," CAR-TR-210, CS-TR-1685, July 1986.

[2] Sharat Chandran and Larry S. Davis, "The Hough Transform on the Butterfly and the NCUBE," CAR-TR-226, CS-TR-1713, September 1986.

[3] Jacqueline J. LeMoigne, "Domain-Dependent Reasoning for Visual Navigation," CAR-TR-230, CS-TR-1721, October, 1986.

[4] Sunil Puri and Larry S. Davis, "Two Dimensional Path Planning with Obstacles and Shadows," CAR-TR-255, CS-TR-1760, January, 1987.

[5] Minoru Asada and Saburo Tsuji, "Shape from Projecting a Stripe Pattern," CAR-TR-263, CS-TR-1773, January, 1987.

[6] Thor Bestul and Larry S. Davis, "On Computing Histograms of Images in $\log n$ Time Using Fat Pyramids," CAR-TR-271, CS-TR-1791, February 1987.

[7] Sharat Chandran, Larry S. Davis, Daniel Dementhon, Sven J. Dickinson, Suresh Gajulapalli, Shie-Rei Huang, Todd R. Kushner, Jacqueline J. LeMoigne, Sunil Puri, Tharakesh Siddalingaiah and Phillip Veatch, CAR-TR-285, CS-TR-1831, April 1987.

[8] Minoru Asada, "3-D Road Structure from Motion Stereo," CAR-TR-286, CS-TR-1839, April 1987.

[9] Shie-Rei Huang and Larry S. Davis, "A Tight Upper Bound for the Speedup of Parallel Best-First Branch-and-Bound Algorithms," CAR-TR-290, CS-TR-1852, May 1987.

[10] Phillip A. Veatch and Larry S. Davis, "Range Imagery Algorithms for the Detection of Obstacles by Autonomous Vehicles," CAR-TR-309, CS-TR-1888, July 1987.

[11] Phillip A. Veatch and Larry S. Davis, "IRS: A Simulator for Autonomous Land Vehicle Navigation," CAR-TR-310, CS-TR-1889, July 1987.

[12] Sven J. Dickinson and Larry S. Davis, "An Expert Vision System for Autonomous Land Vehicle Road Following," CAR-TR-330, CS-TR-1932, October 1987.

[13] Minoru Asada, "Building a 3-D World Model for a Mobile Robot from Sensory Data," CAR-TR-332, CS-TR-1936, October 1987.

[14] Daniel Dementhon, Tharakesh Siddalingaiah and Larry S. Davis, "Production of Dense Range Images With the CVL Light-Stripe Range Scanner," CAR-TR-337, CS-TR-1962, December 1987.

8. FUTURE PLANS

In May 1988, the Computer Vision Laboratory was awarded a DARPA Sponsored ETL contract to continue its research on visual navigation. In this section, we outline our proposed plans for research during the initial stages of this effort.

Our research program is focused on four goals:

1) development of fundamental algorithms for visual navigation,

2) laboratory scale experimentation involving the integration of these algorithms into complete visual navigation systems,

3) development of parallel algorithms for vision, specifically concentrating on the DARPA-developed parallel machines in the Laboratory (the WARP and the Connection Machine),

4) experimental evaluation of our visual navigation algorithms using the Martin Marietta ALV and/or the CMU NavLab.

We describe each of these briefly in the following paragraphs.

Our research on algorithms will be driven by two applications in navigation. The first application attempts to identify a smoothest path for the ALV through a previously identified safe corridor; i.e., we assume that the ALV has identified a navigable corridor, and we would like to exercise fine control of the vehicle's motion through this corridor in order to maximize the smoothness of the ride through the corridor. The motivations for identifying such a path include minimizing sensor rock and providing human occupants of the vehicle, performing

other tasks, with a comfortable ride. Our approach involves analyzing the range data from the ERIM scanner, first transforming it into a uniformly sampled elevation map (which includes a correction for vehicle motion while the raw range data is being collected) and then, based on a model of the vehicle suspension, transforming the elevation map into an abstract "smoothness" map through which an actual path may be plotted.

The second application involves developing and executing a plan of action that would allow a mobile robot to perform some tracking or interception task with respect to a moving object. We call this project RAMBO (Robots Acting on Moving BOdies). We are building a special laboratory facility for RAMBO. The project will involve research in low level vision (on parallel machines), pose and motion estimation for objects of known geometric structure, and motion and task planning.

RAMBO will be developed using a dual robot facility. The first robot, the one used in our previous research on road navigation, will carry a TV camera and a laser pointing device. The second robot (which was purchased with University funds) will carry a target; the surface of the target will contain a set of sensors that can measure the duration of time that they are illuminated by the laser. The second robot will carry the target through a constrained trajectory through space, and the first robot will attempt to track the second in such a way that it can illuminate a given sequence of sensors on the target for specific durations of time. We expect to use the Connection Machine heavily on this project.

We are also studying parallel vision algorithms in a more general way on the Connection Machine. Specifically, we are interested in designing efficient multiresolution image processing systems on the Connection Machine. The basic problem is that many multiresolution (or pyramid) image analysis algorithms utilize one level of the pyramid at a time. If one were simply to map the pixels in the pyramid 1-1 onto processors in the Connection Machine, one would obtain very poor utilization of the machine. We have identified two different approaches towards obtaining better utilization. The first involves using a variable number of processors per pixel, increasing as a function of level in the Connection Machine. The resulting structure is called a Fat Pyramid; we have developed many basic algorithms for Fat Pyramids, and are currently implementing them on the Connection Machine. The second approach, which will probably be more practical, involves making additional copies of the images at each level, with the number of copies increasing with level. Many basic image processing operations can then be decomposed in a way that part of the operation is performed on each copy. The hard problem is being able to use a SIMD machine to compute all of the partial results in parallel. Both the Fat Pyramid and the redundant representation would also be useful in systems based on a focus of attention mechanism, where only windows of a large image are analyzed (such as our road following system).

Finally, we are planning a series of experiments on the ALV over the next few years. During the Spring of 1989 we hope to be able to test our algorithms for identifying smooth paths, and in the Fall of 1989 we would like to experiment

with a system for visual tracking of an object from a moving platform based on the RAMBO project.

END

FILMED

7-89

DTIC